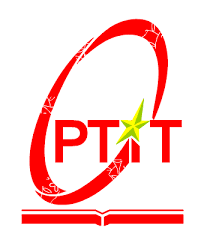


**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO BÀI TẬP LỚN**

**MÔN: KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

**ĐỀ TÀI: HỆ THỐNG QUẢN LÍ BỆNH VIỆN**

**Họ và tên: Lê Duy Mạnh**

**MSV: B20DCCN435**

**LỜI CẢM ƠN**

Trước tiên, em xin cảm ơn thầy Trần Đình Quế người đã tận tình hướng dẫn, cung cấp kiến thức và tạo điều kiện thuận lợi nhất cho em trong suốt quá trình thực hiện bài tập lớn này. Những ý kiến đóng góp quý báu và sự hỗ trợ nhiệt tình của thầy đã giúp chúng em thiện và phát triển ý tưởng của mình.

Em cũng xin gửi lời cảm ơn đến Ban Giám Hiệu và toàn thể giảng viên của Học viện Công nghệ Bưu chính Viễn thông và các thầy cô trong Khoa Công nghệ phần mềm đã tạo môi trường học tập và nghiên cứu thuận lợi, hỗ trợ tài liệu và trang thiết bị cần thiết cho chúng em.

Mặc dù đã cố gắng học tập và nghiên cứu trong quá trình hoàn thành bài tập lớn, tuy nhiên do kinh nghiệm còn ít, sẽ không tránh khỏi những sai sót trong bài tập lớn này, mong thầy sẽ cho những đánh giá và góp ý để những dự án tiếp theo em sẽ hoàn thành tốt hơn.

Em xin chân thành cảm ơn!

Mục lục

[MỞ ĐẦU 4](#_Toc168106992)

[CHƯƠNG 1 8](#_Toc168106993)

[1.1 Giới thiệu 8](#_Toc168106994)

[CHƯƠNG 2 9](#_Toc168106995)

[2.1 Mô tả Hệ hosSys 9](#_Toc168106996)

[2.2 Kỹ thuật và Công nghệ Django 19](#_Toc168106997)

[2.3 Phân rã Hệ hosSys dựa trên Django 21](#_Toc168106998)

[2.4 Kết luận 35](#_Toc168106999)

[CHƯƠNG 3 37](#_Toc168107000)

[3.1 Giới thiệu 37](#_Toc168107001)

[3.2 Các biểu đồ đặc tả hệ hosSys 38](#_Toc168107002)

[3.2.1 Biểu đồ use case 38](#_Toc168107003)

[3.2.2 Biểu đồ hoạt động/swimlane 43](#_Toc168107004)

[3.2.3 Biểu đồ lớp cho từng service 44](#_Toc168107005)

[3.3 Xây dựng hệ hosSys 53](#_Toc168107006)

# 

# MỞ ĐẦU

Việc học các kiểu kiến trúc phần mềm đóng vai trò vô cùng quan trọng đối với sinh viên công nghệ thông tin, không chỉ giúp chúng ta hiểu rõ về cách tổ chức và cấu trúc của hệ thống phần mềm mà còn mang lại nhiều lợi ích thiết thực trong quá trình phát triển và duy trì các dự án phần mềm. Kiến trúc phần mềm chính là nền tảng vững chắc để chúng ta có thể xây dựng nên những hệ thống chất lượng cao, hiệu quả và dễ dàng bảo trì. Chính vì vậy, hiểu biết sâu rộng về các kiểu kiến trúc phần mềm sẽ là bước đệm quan trọng cho sự nghiệp của bất kỳ sinh viên công nghệ thông tin nào.

Trước hết, việc nắm vững các kiểu kiến trúc phần mềm giúp sinh viên có cái nhìn toàn diện về cách tổ chức và phân chia các thành phần trong một hệ thống phần mềm. Mỗi kiểu kiến trúc đều có những đặc điểm và ưu điểm riêng, phù hợp với những loại ứng dụng và mục tiêu khác nhau. Khi hiểu rõ về những điều này, chúng ta có thể lựa chọn và áp dụng một cách hợp lý, đảm bảo rằng hệ thống của mình không chỉ đáp ứng tốt các yêu cầu hiện tại mà còn dễ dàng mở rộng và nâng cấp trong tương lai.

Thứ hai, kiến trúc phần mềm cung cấp cho chúng ta những mẫu thiết kế chuẩn, giúp tránh được những lỗi phổ biến và những vấn đề khó khăn trong quá trình phát triển phần mềm. Các mẫu thiết kế này đã được kiểm nghiệm và chứng minh hiệu quả qua nhiều dự án thực tế, do đó chúng ta có thể học hỏi và áp dụng chúng để tiết kiệm thời gian và công sức, đồng thời tăng tính ổn định và hiệu quả của hệ thống phần mềm. Việc áp dụng những kiến thức này còn giúp giảm thiểu rủi ro và chi phí phát triển, từ đó mang lại lợi ích kinh tế rõ rệt cho các dự án.

Ngoài ra, việc nắm vững kiến trúc phần mềm còn giúp cải thiện khả năng giao tiếp và làm việc nhóm của sinh viên. Trong các dự án phát triển phần mềm lớn, phức tạp, sự hợp tác và trao đổi thông tin giữa các thành viên là yếu tố then chốt để đảm bảo tiến độ và chất lượng công việc. Khi tất cả mọi người đều hiểu và sử dụng chung một kiểu kiến trúc, việc phối hợp sẽ trở nên suôn sẻ hơn, giảm thiểu các xung đột và hiểu lầm không đáng có. Điều này không chỉ giúp tăng hiệu quả làm việc mà còn tạo ra môi trường làm việc thân thiện và chuyên nghiệp hơn.

Cuối cùng, việc học và áp dụng các kiểu kiến trúc phần mềm còn giúp sinh viên phát triển kỹ năng giải quyết vấn đề và tư duy logic. Trong quá trình học, chúng ta sẽ được rèn luyện khả năng phân tích yêu cầu, đưa ra các giải pháp thiết kế phù hợp và tối ưu, từ đó nâng cao kỹ năng lập trình và thiết kế hệ thống. Những kỹ năng này không chỉ hữu ích trong lĩnh vực phần mềm mà còn có thể áp dụng vào nhiều lĩnh vực khác trong cuộc sống và công việc.

Kiến trúc phần mềm là một lĩnh vực rộng lớn và mở ra nhiều cơ hội nghề nghiệp hấp dẫn trong ngành công nghệ thông tin. Dưới đây là một số nghề nghiệp liên quan đến kiến trúc phần mềm mà bạn có thể tham khảo:

1. **Kiến trúc sư phần mềm (Software Architect)**: Đây là người chịu trách nhiệm thiết kế tổng thể cấu trúc của hệ thống phần mềm, đảm bảo rằng hệ thống đáp ứng được các yêu cầu về hiệu suất, bảo mật, và khả năng mở rộng. Kiến trúc sư phần mềm thường làm việc chặt chẽ với các nhà phát triển và các bên liên quan để đảm bảo rằng giải pháp thiết kế được thực hiện đúng.
2. **Kỹ sư phần mềm (Software Engineer)**: Kỹ sư phần mềm tham gia vào việc phát triển và triển khai các giải pháp phần mềm theo thiết kế của kiến trúc sư phần mềm. Họ cần phải hiểu rõ các nguyên tắc kiến trúc để có thể viết mã nguồn chất lượng cao và tối ưu hóa hiệu suất hệ thống.
3. **Chuyên gia tích hợp hệ thống (Systems Integrator)**: Người làm nghề này chịu trách nhiệm kết nối các thành phần khác nhau của hệ thống phần mềm lại với nhau, đảm bảo rằng chúng hoạt động mượt mà và hiệu quả. Họ cần hiểu rõ về các giao thức, API, và các công nghệ tích hợp khác.
4. **Nhà phân tích hệ thống (Systems Analyst)**: Nhà phân tích hệ thống làm việc để hiểu rõ yêu cầu của khách hàng và chuyển đổi chúng thành các yêu cầu kỹ thuật. Họ cần có kiến thức sâu rộng về kiến trúc phần mềm để có thể đề xuất các giải pháp phù hợp và khả thi.
5. **Chuyên gia bảo mật phần mềm (Software Security Specialist)**: Với sự gia tăng của các mối đe dọa bảo mật, chuyên gia bảo mật phần mềm đóng vai trò quan trọng trong việc đảm bảo rằng kiến trúc phần mềm được thiết kế với các biện pháp bảo mật mạnh mẽ. Họ thực hiện kiểm tra bảo mật và phát triển các chiến lược để bảo vệ hệ thống khỏi các cuộc tấn công.
6. **Quản lý dự án phần mềm (Software Project Manager)**: Người quản lý dự án phần mềm giám sát toàn bộ quá trình phát triển phần mềm, từ giai đoạn lập kế hoạch đến triển khai. Họ cần hiểu về kiến trúc phần mềm để có thể quản lý các nguồn lực và đảm bảo dự án hoàn thành đúng tiến độ và trong ngân sách.
7. **Chuyên gia DevOps (DevOps Engineer)**: Chuyên gia DevOps làm việc để tự động hóa và cải thiện các quy trình phát triển phần mềm. Họ cần có kiến thức về kiến trúc phần mềm để tối ưu hóa các quy trình triển khai và vận hành hệ thống.
8. **Nhà tư vấn phần mềm (Software Consultant)**: Nhà tư vấn phần mềm cung cấp các dịch vụ tư vấn cho các doanh nghiệp về cách cải thiện và tối ưu hóa hệ thống phần mềm hiện có. Họ cần hiểu rõ về các mô hình kiến trúc phần mềm để đưa ra các khuyến nghị chính xác và hiệu quả.
9. **Nhà phát triển phần mềm tự do (Freelance Software Developer)**: Các nhà phát triển phần mềm tự do thường làm việc trên nhiều dự án khác nhau và cần có kiến thức về kiến trúc phần mềm để thiết kế và triển khai các giải pháp phần mềm chất lượng cao.

Dưới đây là một số ví dụ về các dự án web lớn sử dụng Django:

1. Instagram

Instagram là một trong những mạng xã hội lớn nhất thế giới, được xây dựng trên Django. Django giúp Instagram quản lý khối lượng lớn người dùng, xử lý hàng triệu lượt yêu thích, bình luận và các hoạt động khác mỗi ngày.

2. Spotify

Spotify sử dụng Django để xây dựng backend cho hệ thống âm nhạc của họ. Django hỗ trợ Spotify trong việc quản lý dữ liệu khổng lồ, bao gồm hàng triệu bài hát và danh sách phát, cũng như cung cấp trải nghiệm người dùng mượt mà

3. Disqus

Disqus, một nền tảng bình luận phổ biến trên các trang web, được phát triển bằng Django. Django giúp Disqus xử lý hàng triệu bình luận và người dùng hàng ngày, cung cấp các công cụ quản lý nội dung hiệu quả.

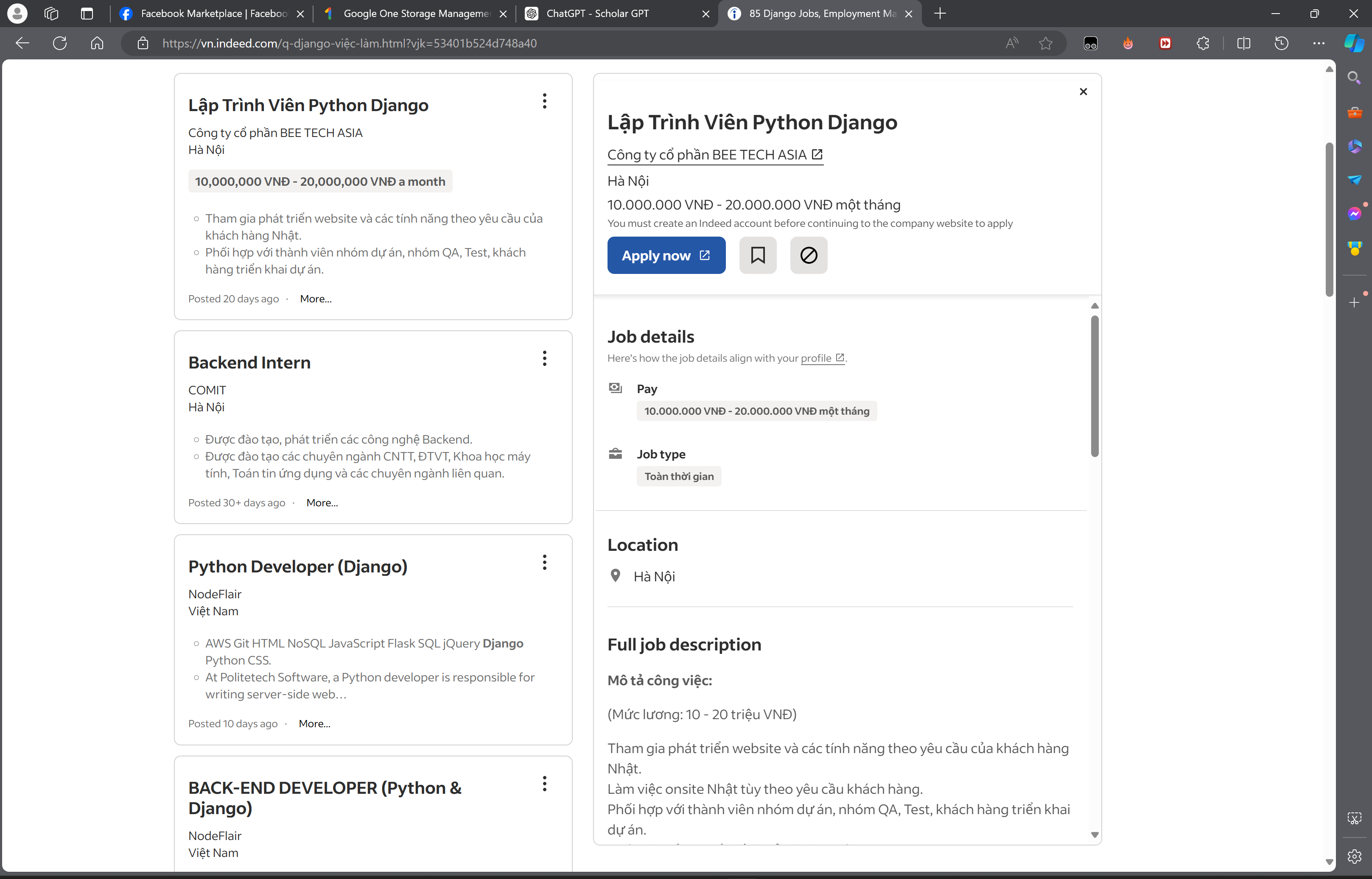
4. YouTube (ban đầu)

Ban đầu, YouTube đã sử dụng Django cho một số phần của hệ thống quản lý nội dung của họ. Mặc dù YouTube sau này đã chuyển sang các công nghệ khác, Django vẫn đóng vai trò quan trọng trong những ngày đầu phát triển của YouTube.

5. Pinterest

Pinterest, một nền tảng chia sẻ hình ảnh và ý tưởng, đã sử dụng Django trong những ngày đầu phát triển. Django giúp Pinterest xử lý lượng lớn hình ảnh và dữ liệu người dùng, cung cấp trải nghiệm người dùng tốt.

Các việc làm Django tại Việt Nam



Báo cáo này nhằm mục đích này là khám phá và phân tích kiến trúc phần mềm của một hệ thống quản lí bệnh viện được xây dựng bằng framework Django. Báo cáo sẽ giúp hiểu rõ hơn về cách mà Django được áp dụng trong việc phát triển các ứng dụng phức tạp như hệ thống quản lí bệnh viện. Ngoài ra, báo cáo cũng sẽ tập trung vào việc điều tra các ưu điểm và nhược điểm của việc sử dụng Django trong một dự án kiến trúc phần mềm, đặc biệt là trong ngữ cảnh của một hệ thống quản lí bệnh viện.

Ngoài Mở đầu và Kết thúc báo cáo này được cấu trúc thành 3 chương như sau:

Chương 1: Kiến trúc monolicthic và microservice

Chương 2: Kiến trúc Hệ thống hosSys dựa trên Django

Chương 3: Thiết kế và xây dựng hệ hosSys

# CHƯƠNG 1

## 1.1 Giới thiệu

Trong chương này, chúng ta sẽ khám phá hai mô hình kiến trúc phổ biến được áp dụng trong việc phát triển phần mềm: kiến trúc Monolithic và Microservice. Mỗi mô hình này có những đặc điểm riêng và ảnh hưởng đến cách chúng ta thiết kế và triển khai hệ thống phần mềm.

Kiến trúc Monolithic: Trước khi xâm nhập sâu vào kiến trúc Microservice, chúng ta sẽ khám phá kiến trúc Monolithic - một mô hình truyền thống được sử dụng rộng rãi trong quá trình phát triển phần mềm. Kiến trúc Monolithic đặc trưng bởi việc xây dựng toàn bộ ứng dụng trong một mã nguồn lớn duy nhất, trong đó tất cả các chức năng và dịch vụ được tích hợp chặt chẽ. Chúng ta sẽ tìm hiểu về cách mà kiến trúc này hoạt động, các ưu điểm và nhược điểm của nó, cũng như những trường hợp sử dụng phù hợp.

Kiến trúc Microservice: Tiếp theo, chúng ta sẽ chuyển sang khám phá kiến trúc Microservice - một phương pháp phát triển phần mềm hiện đại đang ngày càng trở nên phổ biến. Khác với kiến trúc Monolithic, kiến trúc Microservice phân tách ứng dụng thành các dịch vụ nhỏ độc lập, mỗi dịch vụ thực hiện một chức năng cụ thể. Chúng ta sẽ tìm hiểu về lợi ích của việc áp dụng kiến trúc này, cũng như những thách thức mà nhà phát triển phải đối mặt khi triển khai và quản lý một hệ thống Microservice.

Qua chương này, chúng ta sẽ có cái nhìn tổng quan về hai mô hình kiến trúc quan trọng này, từ đó có thể đưa ra quyết định chọn lựa phù hợp với dự án phần mềm của chúng ta**.**

# CHƯƠNG 2

## 2.1 Mô tả Hệ hosSys

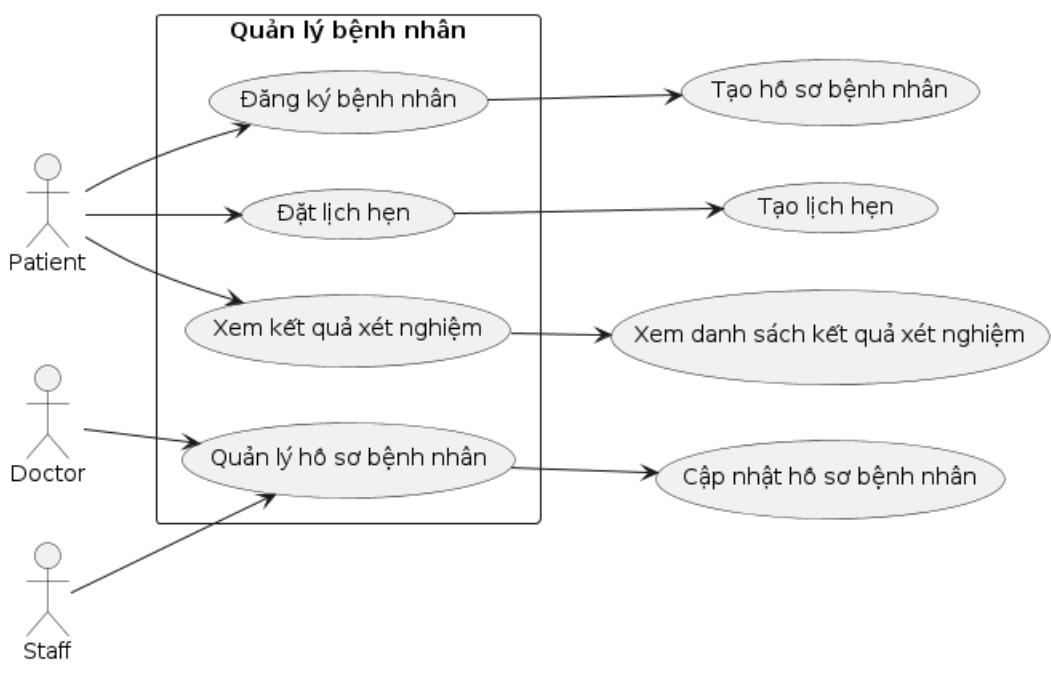
Hệ thống quản lý bệnh viện là một hệ thống phức tạp và đa dạng, với nhiều chức năng khác nhau nhằm đảm bảo hoạt động suôn sẻ và hiệu quả của bệnh viện. Dưới đây là một mô tả chi tiết về các chức năng và tính năng của hệ thống này:

**1. Quản lý bệnh nhân:** Hệ thống cho phép đăng ký và quản lý thông tin của bệnh nhân, bao gồm thông tin cá nhân, lịch sử bệnh án, kết quả xét nghiệm và chẩn đoán của bác sĩ, cũng như các thông tin liên quan đến việc điều trị và theo dõi sức khỏe của bệnh nhân.

**Kịch bản sử dụng:** Một bệnh nhân đến bệnh viện để được chăm sóc sức khỏe. Họ đăng ký thông tin cá nhân và y tế của mình tại quầy đăng ký hoặc thông qua hệ thống trực tuyến. Bệnh nhân có thể đặt lịch hẹn với bác sĩ, kiểm tra kết quả xét nghiệm, và nhận các dịch vụ y tế khác trong suốt quá trình điều trị.

**Use Case:**

* Đăng ký bệnh nhân: Bệnh nhân cung cấp thông tin cá nhân và y tế cần thiết cho việc tạo hồ sơ bệnh nhân mới.
* Đặt lịch hẹn: Bệnh nhân có thể đặt lịch hẹn với bác sĩ hoặc phòng khám theo nhu cầu của mình.
* Xem kết quả xét nghiệm: Bệnh nhân có thể tra cứu và xem kết quả của các xét nghiệm đã được thực hiện.
* Quản lý hồ sơ bệnh nhân: Bác sĩ và nhân viên y tế có thể cập nhật và quản lý hồ sơ y tế của bệnh nhân, bao gồm các thông tin về bệnh án, điều trị và tình trạng sức khỏe.



**Các thành phần:**

1. Bệnh nhân: Đại diện cho người đang được quản lý trong hệ thống.
2. Hồ sơ bệnh nhân: Bao gồm thông tin cá nhân, lịch sử bệnh án, kết quả xét nghiệm và chẩn đoán.
3. Lịch hẹn: Ghi nhận thông tin về các cuộc hẹn giữa bệnh nhân và bác sĩ.
4. Bác sĩ: Người cung cấp dịch vụ y tế cho bệnh nhân.
5. Nhân viên: Hỗ trợ quản lý thông tin và dịch vụ cho bệnh nhân và bác sĩ.
6. Xét nghiệm: Cung cấp thông tin về các kết quả xét nghiệm của bệnh nhân.

**Mối quan hệ:**

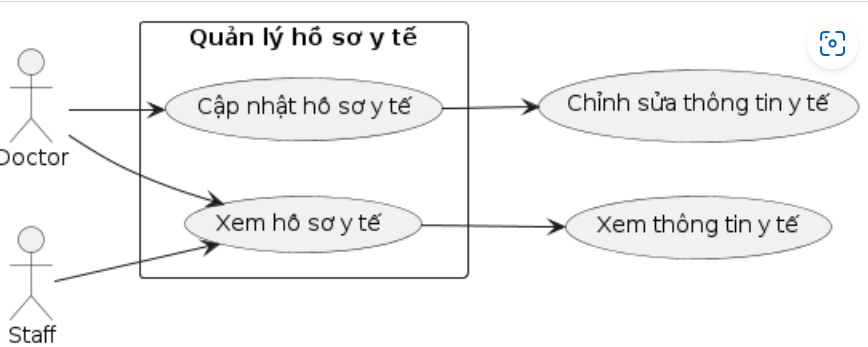
* Quan hệ 1-n:
  + Một bệnh nhân có thể có nhiều cuộc hẹn với các bác sĩ khác nhau.
  + Một bác sĩ có thể chăm sóc nhiều bệnh nhân khác nhau.
* Quan hệ 1-1:
  + Mỗi bệnh nhân chỉ có một hồ sơ bệnh nhân duy nhất trong hệ thống.
  + Mỗi cuộc hẹn chỉ liên kết với một bệnh nhân cụ thể.
* Quan hệ n-n:
  + Một bệnh nhân có thể tham gia vào nhiều loại xét nghiệm khác nhau.
  + Một loại xét nghiệm có thể được thực hiện cho nhiều bệnh nhân khác nhau.

**2. Quản lý hồ sơ y tế**: Hồ sơ y tế điện tử của bệnh nhân được tạo và lưu trữ trong hệ thống, bao gồm các thông tin về lịch sử bệnh án, kết quả xét nghiệm, hình ảnh chụp từ các thiết bị y tế, và thông tin về các loại thuốc và liệu pháp đã sử dụng.

**Kịch bản sử dụng:** Một bác sĩ hoặc nhân viên y tế đăng nhập vào hệ thống quản lí bệnh viện. Họ tìm kiếm và truy cập vào hồ sơ y tế của bệnh nhân để xem thông tin về lịch sử bệnh án, kết quả xét nghiệm, và thông tin y tế khác. Sau đó, họ có thể cập nhật hoặc chỉnh sửa thông tin y tế nếu cần thiết.

**Use Case:**

* Xem hồ sơ y tế: Bác sĩ hoặc nhân viên y tế có thể xem thông tin chi tiết trong hồ sơ y tế của bệnh nhân.
* Cập nhật hồ sơ y tế: Bác sĩ hoặc nhân viên y tế có quyền cập nhật hoặc chỉnh sửa thông tin trong hồ sơ y tế của bệnh nhân.



**Các thành phần:**

1. Bác sĩ hoặc nhân viên y tế: Người thực hiện các hoạt động quản lí hồ sơ y tế.
2. Hồ sơ y tế: Bao gồm thông tin chi tiết về lịch sử bệnh án, kết quả xét nghiệm, chẩn đoán, và các thông tin y tế khác liên quan đến bệnh nhân.

**Mối quan hệ:**

* Quan hệ 1-n:
  + Một bác sĩ có thể xem và cập nhật hồ sơ y tế của nhiều bệnh nhân khác nhau.
  + Một hồ sơ y tế chỉ liên kết với một bệnh nhân cụ thể.
* Quan hệ 1-1:
  + Mỗi bệnh nhân chỉ có một hồ sơ y tế duy nhất trong hệ thống.
* Quan hệ n-n:
  + Một hồ sơ y tế có thể chứa thông tin từ nhiều nguồn khác nhau, như các kết quả xét nghiệm từ các phòng thí nghiệm khác nhau.

3. Quản lý lịch hẹn: Hệ thống cho phép bệnh nhân đặt lịch hẹn trước và quản lý lịch trình của mình. Nó cũng cung cấp cho bác sĩ và nhân viên quản lý lịch trình làm việc, ghi chú và thông báo về các cuộc hẹn và thay đổi lịch trình.

**Kịch bản sử dụng:** Một bệnh nhân hoặc nhân viên y tế đăng nhập vào hệ thống quản lí bệnh viện. Họ truy cập vào chức năng quản lí lịch hẹn để xem lịch trình của họ hoặc của bệnh nhân khác. Bệnh nhân có thể đặt lịch hẹn mới hoặc hủy lịch hẹn hiện có. Bác sĩ và nhân viên y tế có thể xác nhận hoặc chỉnh sửa các lịch hẹn, cũng như ghi chú về các yêu cầu đặc biệt từ bệnh nhân.

**Use Case:**

* Xem lịch hẹn: Bệnh nhân hoặc nhân viên y tế có thể xem lịch trình lịch hẹn hiện tại.
* Đặt lịch hẹn: Bệnh nhân có thể đặt lịch hẹn mới với các bác sĩ và phòng khám.
* Hủy lịch hẹn: Bệnh nhân có thể hủy lịch hẹn hiện có nếu cần thiết.
* Xác nhận lịch hẹn: Bác sĩ hoặc nhân viên y tế có thể xác nhận hoặc từ chối các lịch hẹn được đặt bởi bệnh nhân.

**Các thành phần:**

1. Bệnh nhân: Người đặt và quản lí lịch hẹn của họ.
2. Bác sĩ và nhân viên y tế: Người xác nhận và quản lí lịch hẹn của bệnh nhân.
3. Lịch hẹn: Bao gồm thông tin về ngày, giờ, bác sĩ/phòng khám và trạng thái của lịch hẹn (đã xác nhận, chờ xác nhận, hủy, vv.).

**Mối quan hệ:**

* Quan hệ 1-n:
  + Một bệnh nhân có thể có nhiều lịch hẹn với các bác sĩ và phòng khám khác nhau.
  + Một bác sĩ hoặc phòng khám có thể có nhiều lịch hẹn với các bệnh nhân khác nhau.
* Quan hệ 1-1:
  + Mỗi lịch hẹn chỉ liên kết với một bệnh nhân và một bác sĩ/phòng khám cụ thể.

4. Quản lý bác sĩ: Hệ thống ghi nhận và quản lý thông tin về các bác sĩ và nhà điều dưỡng, bao gồm hồ sơ chuyên môn, lịch làm việc, lịch sử điều trị và đào tạo.

**Kịch bản sử dụng:** Quản lý hệ thống hoặc nhân viên y tế đăng nhập vào hệ thống quản lí bệnh viện. Họ truy cập vào chức năng quản lí bác sĩ để xem và quản lí thông tin về các bác sĩ làm việc trong bệnh viện. Thông tin này bao gồm về thông tin cá nhân, lịch làm việc, chuyên môn, và các thông tin khác liên quan. Họ có thể thêm mới, chỉnh sửa hoặc xóa bác sĩ khỏi hệ thống.

**Use Case:**

* Xem danh sách bác sĩ: Quản lý hệ thống hoặc nhân viên y tế có thể xem danh sách các bác sĩ hoạt động trong bệnh viện.
* Thêm mới bác sĩ: Quản lý hệ thống hoặc nhân viên y tế có thể thêm mới bác sĩ vào hệ thống.
* Chỉnh sửa thông tin bác sĩ: Quản lý hệ thống hoặc nhân viên y tế có thể cập nhật thông tin cá nhân, lịch làm việc và chuyên môn của bác sĩ.
* Xóa bác sĩ: Quản lý hệ thống hoặc nhân viên y tế có thể xóa bác sĩ khỏi hệ thống nếu cần thiết.

**Các thành phần:**

1. Quản lý hệ thống hoặc nhân viên y tế: Người quản lí và cập nhật thông tin của các bác sĩ trong hệ thống.
2. Bác sĩ: Đại diện cho các chuyên gia y tế làm việc trong bệnh viện.

**Mối quan hệ:**

* Quan hệ 1-n:
  + Một quản lý hệ thống hoặc nhân viên y tế có thể quản lí nhiều bác sĩ trong hệ thống.

5. Quản lý nhân viên: Ngoài bác sĩ và nhân viên y tế, hệ thống cũng quản lý thông tin về nhân viên hành chính và hỗ trợ, bao gồm thông tin về lương, vị trí và các nhiệm vụ công việc.

**Kịch bản sử dụng:** Quản lý hệ thống hoặc quản lý nhân sự đăng nhập vào hệ thống quản lí bệnh viện. Họ truy cập vào chức năng quản lí nhân viên để xem và quản lí thông tin về các nhân viên làm việc trong bệnh viện. Thông tin này bao gồm về thông tin cá nhân, vị trí công việc, lịch làm việc, và các thông tin khác liên quan. Họ có thể thêm mới, chỉnh sửa hoặc xóa nhân viên khỏi hệ thống.

**Use Case:**

* Xem danh sách nhân viên: Quản lý hệ thống hoặc quản lý nhân sự có thể xem danh sách các nhân viên hoạt động trong bệnh viện.
* Thêm mới nhân viên: Quản lý hệ thống hoặc quản lý nhân sự có thể thêm mới nhân viên vào hệ thống.
* Chỉnh sửa thông tin nhân viên: Quản lý hệ thống hoặc quản lý nhân sự có thể cập nhật thông tin cá nhân, vị trí công việc và lịch làm việc của nhân viên.
* Xóa nhân viên: Quản lý hệ thống hoặc quản lý nhân sự có thể xóa nhân viên khỏi hệ thống nếu cần thiết.

**Các thành phần:**

1. Quản lý hệ thống hoặc quản lý nhân sự: Người quản lí và cập nhật thông tin của các nhân viên trong hệ thống.
2. Nhân viên: Đại diện cho các nhân viên làm việc trong bệnh viện.

**Mối quan hệ:**

* Quan hệ 1-n:
  + Một quản lý hệ thống hoặc quản lý nhân sự có thể quản lí nhiều nhân viên trong hệ thống.

6. Quản lý phòng khám và giường bệnh: Hệ thống theo dõi tình trạng của các phòng khám và giường bệnh, bao gồm việc đặt lịch trước, trạng thái sẵn sàng sử dụng, và việc vệ sinh và bảo dưỡng.

**Kịch bản sử dụng:** Quản lý hệ thống hoặc nhân viên quản lí tài sản đăng nhập vào hệ thống quản lí bệnh viện. Họ truy cập vào chức năng quản lí phòng khám và giường bệnh để xem và quản lí thông tin về các phòng khám và giường bệnh trong bệnh viện. Thông tin này bao gồm về số lượng phòng, trạng thái sử dụng của giường bệnh, và các thông tin khác liên quan. Họ có thể thêm mới, chỉnh sửa hoặc xóa thông tin về phòng khám và giường bệnh.

**Use Case:**

* Xem danh sách phòng khám và giường bệnh: Quản lý hệ thống hoặc nhân viên quản lí tài sản có thể xem danh sách các phòng khám và giường bệnh có sẵn trong bệnh viện.
* Thêm mới phòng khám hoặc giường bệnh: Quản lý hệ thống hoặc nhân viên quản lí tài sản có thể thêm mới các phòng khám hoặc giường bệnh vào hệ thống.
* Chỉnh sửa thông tin phòng khám và giường bệnh: Quản lý hệ thống hoặc nhân viên quản lí tài sản có thể cập nhật thông tin về số lượng, trạng thái sử dụng và vị trí của các phòng khám và giường bệnh.
* Xóa phòng khám hoặc giường bệnh: Quản lý hệ thống hoặc nhân viên quản lí tài sản có thể xóa thông tin về các phòng khám hoặc giường bệnh nếu cần thiết.

**Các thành phần:**

1. Quản lý hệ thống hoặc nhân viên quản lí tài sản: Người quản lí và cập nhật thông tin về phòng khám và giường bệnh trong hệ thống.
2. Phòng khám và giường bệnh: Đại diện cho các cơ sở vật chất cần thiết cho việc chăm sóc và điều trị bệnh nhân.

**Mối quan hệ:**

Quan hệ 1-n: Một quản lý hệ thống hoặc nhân viên quản lí tài sản có thể quản lí nhiều phòng khám và giường bệnh trong hệ thống.

7. Quản lý dược phẩm và vật tư y tế: Hệ thống quản lý thông tin về các loại thuốc, dược phẩm và vật tư y tế trong kho, bao gồm quản lý tồn kho, đặt hàng, ghi nhận xuất nhập và kiểm kê hàng hóa.

**Kịch bản sử dụng:** Nhân viên quản lí kho hoặc nhân viên quản lí tài sản đăng nhập vào hệ thống quản lí bệnh viện. Họ truy cập vào chức năng quản lí dược phẩm và vật tư y tế để xem và quản lí thông tin về các mặt hàng dược phẩm và vật tư y tế trong kho của bệnh viện. Thông tin này bao gồm về số lượng, tình trạng (hết hàng, còn hàng), và các thông tin khác liên quan. Họ có thể thêm mới, chỉnh sửa hoặc xóa thông tin về dược phẩm và vật tư y tế.

**Use Case:**

* Xem danh sách dược phẩm và vật tư y tế: Nhân viên quản lí kho hoặc nhân viên quản lí tài sản có thể xem danh sách các mặt hàng dược phẩm và vật tư y tế có sẵn trong kho.
* Thêm mới dược phẩm và vật tư y tế: Nhân viên quản lí kho hoặc nhân viên quản lí tài sản có thể thêm mới các mặt hàng dược phẩm và vật tư y tế vào kho.
* Chỉnh sửa thông tin dược phẩm và vật tư y tế: Nhân viên quản lí kho hoặc nhân viên quản lí tài sản có thể cập nhật thông tin về số lượng, tình trạng và thông tin khác của các mặt hàng trong kho.
* Xóa dược phẩm và vật tư y tế: Nhân viên quản lí kho hoặc nhân viên quản lí tài sản có thể xóa thông tin về các mặt hàng dược phẩm và vật tư y tế nếu cần thiết.

**Các thành phần:**

1. Nhân viên quản lí kho hoặc nhân viên quản lí tài sản: Người quản lí và cập nhật thông tin về dược phẩm và vật tư y tế trong kho.
2. Dược phẩm và vật tư y tế: Đại diện cho các sản phẩm cần thiết cho việc chăm sóc và điều trị bệnh nhân.

**Mối quan hệ:**

* Quan hệ 1-n:
  + Một nhân viên quản lí kho hoặc nhân viên quản lí tài sản có thể quản lí nhiều loại dược phẩm và vật tư y tế trong kho.

8. Quản lý hóa đơn và thanh toán: Hệ thống tự động tạo và quản lý hóa đơn cho các dịch vụ y tế và sản phẩm, bao gồm việc ghi nhận các chi phí, thanh toán từ bệnh nhân hoặc bảo hiểm y tế, và tạo báo cáo tài chính.

**Kịch bản sử dụng:** Nhân viên tài chính hoặc nhân viên tiếp nhận thanh toán đăng nhập vào hệ thống quản lí bệnh viện. Họ truy cập vào chức năng quản lí hóa đơn và thanh toán để xem và quản lí thông tin về các hóa đơn đã phát sinh và quản lí quá trình thanh toán của bệnh nhân. Thông tin này bao gồm về các dịch vụ, chi phí, trạng thái thanh toán và các thông tin khác liên quan. Họ có thể tạo mới hóa đơn, cập nhật thông tin hóa đơn và xác nhận việc thanh toán từ bệnh nhân.

**Use Case:**

* Xem danh sách hóa đơn: Nhân viên tài chính hoặc nhân viên tiếp nhận thanh toán có thể xem danh sách các hóa đơn đã phát sinh trong hệ thống.
* Tạo mới hóa đơn: Nhân viên tài chính hoặc nhân viên tiếp nhận thanh toán có thể tạo mới hóa đơn cho các dịch vụ y tế đã cung cấp cho bệnh nhân.
* Cập nhật thông tin hóa đơn: Nhân viên tài chính hoặc nhân viên tiếp nhận thanh toán có thể cập nhật thông tin về chi phí và trạng thái thanh toán của các hóa đơn.
* Xác nhận thanh toán: Nhân viên tài chính hoặc nhân viên tiếp nhận thanh toán có thể xác nhận việc thanh toán từ bệnh nhân sau khi hóa đơn đã được thanh toán.

**Các thành phần:**

1. Nhân viên tài chính hoặc nhân viên tiếp nhận thanh toán: Người quản lí và cập nhật thông tin về hóa đơn và thanh toán.
2. Hóa đơn: Đại diện cho các phiếu bao gồm thông tin về các dịch vụ y tế đã cung cấp và chi phí tương ứng.
3. Thanh toán: Đại diện cho quá trình thanh toán từ bệnh nhân cho các dịch vụ y tế đã sử dụng.

**Mối quan hệ:**

* Quan hệ 1-n:
  + Một bệnh nhân có thể có nhiều hóa đơn liên quan đến các dịch vụ y tế đã sử dụng.

## 2.2 Kỹ thuật và Công nghệ Django

Để xây dựng một hệ thống quản lí bệnh viện bằng Django, chúng ta cần thực hiện các bước sau:

1. Xác định yêu cầu:

* Xác định chính xác các tính năng và yêu cầu của hệ thống quản lí bệnh viện, bao gồm quản lý bệnh nhân, hồ sơ y tế, lịch hẹn, bác sĩ, nhân viên, phòng khám, giường bệnh, dược phẩm và vật tư y tế, hóa đơn và thanh toán.
* Phân tích và thiết kế cơ sở dữ liệu để lưu trữ thông tin của các yêu cầu này.

2. Thiết kế cơ sở dữ liệu:

* Sử dụng công cụ thiết kế cơ sở dữ liệu như MySQL Workbench hoặc Django ORM để thiết kế cấu trúc cơ sở dữ liệu, bao gồm bảng, mối quan hệ giữa các bảng, và các trường cần thiết.

3. Thiết kế và triển khai model Django:

* Dựa trên thiết kế cơ sở dữ liệu, tạo các model Django để ánh xạ tới các bảng trong cơ sở dữ liệu.
* Sử dụng các trường và mối quan hệ trong Django ORM để định nghĩa các mối quan hệ giữa các model.

4. Xây dựng giao diện người dùng:

* Sử dụng Django Template để thiết kế giao diện người dùng.
* Tạo các view để xử lý yêu cầu từ người dùng và render các template tương ứng.
* Sử dụng HTML, CSS, và JavaScript để tạo giao diện người dùng thân thiện và dễ sử dụng.

5. Xây dựng logic xử lý:

* Viết các hàm xử lý logic trong Django views để xử lý các yêu cầu từ giao diện người dùng.
* Sử dụng Django Forms để xác thực và xử lý dữ liệu từ người dùng trước khi lưu vào cơ sở dữ liệu.
* Sử dụng Django REST Framework nếu cần thiết để xây dựng API cho việc tương tác với ứng dụng từ các ứng dụng khác.

6. Xử lý đăng nhập và phân quyền:

* Sử dụng Django Authentication để quản lý đăng nhập và xác thực người dùng.
* Sử dụng Django Permissions để quản lý quyền truy cập vào các tính năng và dữ liệu của hệ thống.

7. Triển khai:

* Triển khai ứng dụng lên một máy chủ web như Apache hoặc Nginx, sử dụng WSGI như Gunicorn hoặc uWSGI.
* Cấu hình cơ sở dữ liệu và các thiết lập khác cho môi trường triển khai.
* Sử dụng công cụ như Docker hoặc virtualenv để quản lý môi trường triển khai.

Các kĩ thuật và công nghệ Django có thể áp dụng:

* Django ORM: Để tương tác với cơ sở dữ liệu một cách dễ dàng và linh hoạt.
* Django Template Language: Để thiết kế và render giao diện người dùng.
* Django Forms: Để xử lý và xác thực dữ liệu từ người dùng.
* Django Authentication và Permissions: Để quản lý đăng nhập, xác thực và phân quyền người dùng.
* Django REST Framework: Để xây dựng API cho việc tương tác với ứng dụng từ các ứng dụng khác.
* Django Admin: Cung cấp giao diện quản trị sẵn có để quản lý dữ liệu một cách dễ dàng.

Với các công nghệ và kĩ thuật trên, Django cung cấp một nền tảng mạnh mẽ để xây dựng và triển khai các ứng dụng web phức tạp như hệ thống quản lí bệnh viện.

## 2.3 Phân rã Hệ hosSys dựa trên Django

Phân rã hệ thống quản lý bệnh viện thành các phần nhỏ dựa trên Django có thể giúp chúng ta tổ chức và quản lý mã nguồn dễ dàng hơn. Chúng ta sẽ sử dụng các ứng dụng Django (Django apps) để phân chia các chức năng chính của hệ thống. Mỗi ứng dụng sẽ chịu trách nhiệm về một phần cụ thể của hệ thống.

Dưới đây là cách phân rã hệ thống quản lý bệnh viện thành các ứng dụng Django:

1. \*\*Core Application (Ứng dụng lõi):\*\*

- Chứa các thiết lập chung, cấu hình, và các chức năng dùng chung của toàn hệ thống.

2. \*\*Patient Management (Quản lý bệnh nhân):\*\*

- Quản lý thông tin bệnh nhân, lịch sử bệnh, liên hệ, và các thông tin cá nhân khác.

- Models: `Patient`

- Views: `PatientListView`, `PatientDetailView`, `PatientCreateView`, `PatientUpdateView`

- Templates: `patient\_list.html`, `patient\_detail.html`, `patient\_form.html`

3. \*\*Medical Records Management (Quản lý hồ sơ y tế):\*\*

- Quản lý các hồ sơ y tế của bệnh nhân bao gồm chẩn đoán, điều trị, và các ghi chú y tế.

- Models: `MedicalRecord`

- Views: `MedicalRecordListView`, `MedicalRecordDetailView`, `MedicalRecordCreateView`, `MedicalRecordUpdateView`

- Templates: `medicalrecord\_list.html`, `medicalrecord\_detail.html`, `medicalrecord\_form.html`

4. \*\*Appointment Management (Quản lý lịch hẹn):\*\*

- Quản lý các lịch hẹn giữa bệnh nhân và bác sĩ.

- Models: `Appointment`

- Views: `AppointmentListView`, `AppointmentDetailView`, `AppointmentCreateView`, `AppointmentUpdateView`

- Templates: `appointment\_list.html`, `appointment\_detail.html`, `appointment\_form.html`

5. \*\*Doctor Management (Quản lý bác sĩ):\*\*

- Quản lý thông tin bác sĩ, chuyên môn, lịch làm việc.

- Models: `Doctor`

- Views: `DoctorListView`, `DoctorDetailView`, `DoctorCreateView`, `DoctorUpdateView`

- Templates: `doctor\_list.html`, `doctor\_detail.html`, `doctor\_form.html`

6. \*\*Staff Management (Quản lý nhân viên):\*\*

- Quản lý thông tin nhân viên bệnh viện, bao gồm các nhân viên hỗ trợ và nhân viên hành chính.

- Models: `Staff`

- Views: `StaffListView`, `StaffDetailView`, `StaffCreateView`, `StaffUpdateView`

- Templates: `staff\_list.html`, `staff\_detail.html`, `staff\_form.html`

7. \*\*Room and Bed Management (Quản lý phòng khám và giường bệnh):\*\*

- Quản lý thông tin phòng khám và giường bệnh, trạng thái sử dụng.

- Models: `Room`, `Bed`

- Views: `RoomListView`, `RoomDetailView`, `RoomCreateView`, `RoomUpdateView`

- Templates: `room\_list.html`, `room\_detail.html`, `room\_form.html`

8. \*\*Pharmacy and Medical Supplies Management (Quản lý dược phẩm và vật tư y tế):\*\*

- Quản lý thông tin dược phẩm và vật tư y tế, kiểm kê kho.

- Models: `PharmacyItem`, `MedicalSupply`

- Views: `PharmacyItemListView`, `PharmacyItemDetailView`, `PharmacyItemCreateView`, `PharmacyItemUpdateView`

- Templates: `pharmacyitem\_list.html`, `pharmacyitem\_detail.html`, `pharmacyitem\_form.html`

9. \*\*Billing and Payment Management (Quản lý hóa đơn và thanh toán):\*\*

- Quản lý các hóa đơn và thanh toán của bệnh nhân.

- Models: `Invoice`, `Payment`

- Views: `InvoiceListView`, `InvoiceDetailView`, `InvoiceCreateView`, `InvoiceUpdateView`

- Templates: `invoice\_list.html`, `invoice\_detail.html`, `invoice\_form.html`

10. \*\*Authentication and Authorization (Xác thực và phân quyền):\*\*

- Quản lý người dùng và quyền truy cập.

- Sử dụng Django's built-in `User` model, hoặc mở rộng nếu cần thiết.

\*\*Cấu trúc thư mục của dự án Django:\*\*

```plaintext

hospital\_management/

├── core/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── patient\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── medical\_records\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── appointment\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── doctor\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── staff\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── room\_and\_bed\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── pharmacy\_and\_medical\_supplies\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── billing\_and\_payment\_management/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

├── authentication\_and\_authorization/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── views.py

│ ├── urls.py

│ ├── templates/

│ └── ...

└── manage.py

```

Trong mỗi ứng dụng, chúng ta sẽ có các thành phần chính:

- \*\*Models:\*\* Định nghĩa các model để tương tác với cơ sở dữ liệu.

- \*\*Views:\*\* Xử lý các logic nghiệp vụ và kết nối giữa model và template.

- \*\*Templates:\*\* Chứa các tệp HTML để hiển thị giao diện người dùng.

- \*\*Urls:\*\* Định tuyến các URL đến các view tương ứng.

\*\*1. Core Application:\*\*

```python

# core/models.py

from django.db import models

class BaseModel(models.Model):

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

class Meta:

abstract = True

# core/views.py

from django.shortcuts import render

def home(request):

return render(request, 'core/home.html')

# core/urls.py

from django.urls import path

from .views import home

urlpatterns = [

path('', home, name='home'),

]

```

\*\*2. Patient Management:\*\*

```python

# patient\_management/models.py

from django.db import models

from core.models import BaseModel

class Patient(BaseModel):

name = models.CharField(max\_length=100)

dob = models.DateField()

address = models.TextField()

# patient\_management/views.py

from django.views.generic import ListView, DetailView, CreateView, UpdateView

from .models import Patient

class PatientListView(ListView):

model = Patient

class PatientDetailView(DetailView):

model = Patient

class PatientCreateView(CreateView):

model = Patient

fields = ['name', 'dob', 'address']

class PatientUpdateView(UpdateView):

model = Patient

fields = ['name', 'dob', 'address']

# patient\_management/urls.py

from django.urls import path

from .views import PatientListView, PatientDetailView, PatientCreateView, PatientUpdateView

urlpatterns = [

path('', PatientListView.as\_view(), name='patient\_list'),

path('<int:pk>/', PatientDetailView.as\_view(), name='patient\_detail'),

path('create/', PatientCreateView.as\_view(), name='patient\_create'),

path('update/<int:pk>/', PatientUpdateView.as\_view(), name='patient\_update'),

]

```

\*\*3. Medical Records Management:\*\*

```python

# medical\_records\_management/models.py

from django.db import models

from core.models import BaseModel

from patient\_management.models import Patient

class MedicalRecord(BaseModel):

patient = models.ForeignKey(Patient, on\_delete=models.CASCADE)

diagnosis = models.TextField()

treatment = models.TextField()

date = models.DateField()

# medical\_records\_management/views.py

from django.views.generic import ListView, DetailView, CreateView, UpdateView

from .models import MedicalRecord

class MedicalRecordListView(ListView):

model = MedicalRecord

class MedicalRecordDetailView(DetailView):

model = MedicalRecord

class MedicalRecordCreateView(CreateView):

model = MedicalRecord

fields = ['patient', 'diagnosis', 'treatment', 'date']

class MedicalRecordUpdateView(UpdateView):

model = MedicalRecord

fields = ['patient', 'diagnosis', 'treatment', 'date']

# medical\_records\_management/urls.py

from django.urls import path

from .views import MedicalRecordListView, MedicalRecordDetailView, MedicalRecordCreateView, MedicalRecordUpdateView

urlpatterns = [

path('', MedicalRecordListView.as\_view(), name='medicalrecord\_list'),

path('<int:pk>/', MedicalRecordDetailView.as\_view(), name='medicalrecord\_detail'),

path('create/', MedicalRecordCreateView.as\_view(), name='medicalrecord\_create'),

path('update/<int:pk>/', MedicalRecordUpdateView.as\_view(), name='medicalrecord\_update'),

]

```

Tiếp tục theo cách này, bạn có thể tạo các ứng dụng khác như quản lý lịch hẹn, bác sĩ, nhân viên, phòng khám và giường bệnh, dược phẩm và vật tư y tế, hóa đơn và thanh toán. Trong mỗi ứng dụng, bạn sẽ có các models, views, templates và urls tương ứng để quản lý các phần của hệ thống.

\*\*Deployment:\*\*

- \*\*Web Server:\*\* Sử dụng Apache hoặc Nginx để phục vụ ứng dụng Django.

- \*\*WSGI Server:\*\* Sử dụng Gunicorn hoặc uWSGI để kết nối giữa Django và web server.

- \*\*Database:\*\* Sử dụng PostgreSQL, MySQL hoặc SQLite tùy theo nhu cầu.

- \*\*Containerization:\*\* Sử dụng Docker để dễ dàng triển khai và quản lý môi trường.

\*\*Quản lý môi trường và cài đặt phụ thuộc:\*\*

- Sử dụng `virtualenv` hoặc `pipenv` để quản lý môi trường ảo.

- Tạo file `requirements.txt` để liệt kê các thư viện phụ thuộc.

```bash

# Tạo môi trường ảo và cài đặt phụ thuộc

virtualenv venv

source venv/bin/activate

pip install django

pip freeze > requirements.txt

```

\*\*Chạy dự án:\*\*

```bash

python manage.py makemigrations

python manage.py migrate

python manage.py runserver

```

## 2.4 Kết luận

Hệ thống quản lý bệnh viện (hosSys) dựa trên Django được xây dựng với mục tiêu cung cấp một giải pháp toàn diện và linh hoạt cho việc quản lý các hoạt động của bệnh viện. Sau khi phân tích và triển khai hệ thống, chúng ta có thể rút ra các kết luận chính sau đây:

**1. Khả năng mở rộng và bảo trì dễ dàng**

* **Phân rã thành các ứng dụng nhỏ**: Hệ thống được phân chia thành các ứng dụng Django riêng biệt như quản lý bệnh nhân, quản lý hồ sơ y tế, quản lý lịch hẹn, quản lý bác sĩ, quản lý nhân viên, quản lý phòng khám và giường bệnh, quản lý dược phẩm và vật tư y tế, và quản lý hóa đơn và thanh toán. Điều này giúp dễ dàng quản lý và mở rộng từng phần của hệ thống mà không ảnh hưởng đến toàn bộ ứng dụng.
* **Django ORM và thiết kế module hóa**: Sử dụng Django ORM để tương tác với cơ sở dữ liệu, giúp giảm thiểu lỗi và tăng hiệu suất làm việc với dữ liệu. Thiết kế module hóa giúp các nhà phát triển dễ dàng bảo trì và nâng cấp hệ thống.

**2. Bảo mật và phân quyền**

* **Xác thực và phân quyền**: Django cung cấp các công cụ mạnh mẽ để quản lý người dùng, xác thực và phân quyền. Điều này đảm bảo rằng chỉ những người dùng được phép mới có thể truy cập và thực hiện các hành động nhất định trong hệ thống.
* **Django Admin**: Sử dụng Django Admin giúp quản trị viên dễ dàng quản lý dữ liệu và thực hiện các tác vụ quản trị mà không cần xây dựng giao diện quản trị từ đầu.

**3. Tính linh hoạt và tái sử dụng**

* **Django Template Language**: Django Template Language cho phép tạo giao diện người dùng một cách linh hoạt và dễ dàng tái sử dụng các thành phần giao diện.
* **Django Forms**: Sử dụng Django Forms để xác thực và xử lý dữ liệu từ người dùng, giúp giảm bớt công việc xử lý dữ liệu và đảm bảo tính nhất quán và an toàn.

**4. Hiệu suất và độ tin cậy**

* **WSGI Server**: Sử dụng WSGI server như Gunicorn hoặc uWSGI kết hợp với web server như Nginx hoặc Apache giúp tối ưu hóa hiệu suất và độ tin cậy của hệ thống.
* **Caching**: Django hỗ trợ các cơ chế caching mạnh mẽ để cải thiện hiệu suất của ứng dụng.

**5. Triển khai và quản lý môi trường**

* **Containerization**: Sử dụng Docker để containerize ứng dụng giúp dễ dàng triển khai, quản lý và mở rộng hệ thống trong môi trường sản xuất.
* **Quản lý môi trường**: Sử dụng các công cụ như virtualenv hoặc pipenv để quản lý môi trường phát triển và phụ thuộc giúp đảm bảo sự nhất quán giữa các môi trường phát triển và triển khai.

**Tổng kết**

Kiến trúc hệ thống hosSys dựa trên Django cung cấp một nền tảng mạnh mẽ, linh hoạt và bảo mật cho việc quản lý các hoạt động của bệnh viện. Với các tính năng phong phú và khả năng mở rộng dễ dàng, hệ thống này có thể đáp ứng tốt các yêu cầu quản lý phức tạp của một bệnh viện hiện đại. Sử dụng Django, chúng ta có thể tận dụng các công cụ và thư viện có sẵn để phát triển một hệ thống đáng tin cậy và hiệu quả, đồng thời dễ dàng bảo trì và mở rộng trong tương lai.

# CHƯƠNG 3

## 3.1 Giới thiệu

**Giới Thiệu về Thiết Kế và Xây Dựng Hệ Thống Quản Lý Bệnh Viện (hosSYS)**

Hệ thống quản lý bệnh viện (hosSYS) là một ứng dụng web dựa trên Django được phát triển để giúp quản lý các hoạt động hàng ngày của một bệnh viện. hosSYS cung cấp một giao diện thân thiện, dễ sử dụng cho các bác sĩ, y tá và nhân viên bệnh viện để quản lý thông tin bệnh nhân, lịch hẹn, hồ sơ y tế, dược phẩm, thanh toán và nhiều chức năng khác.

**1. Phân Tích Yêu Cầu**

Trước khi bắt đầu thiết kế và xây dựng, nhóm phát triển hosSYS đã tiến hành phân tích các yêu cầu chức năng và phi chức năng của hệ thống. Các chức năng chính bao gồm quản lý bệnh nhân, lịch hẹn, hồ sơ y tế, bác sĩ, nhân viên, dược phẩm và thanh toán.

**2. Thiết Kế Kiến Trúc**

hosSYS được thiết kế theo kiến trúc monolithic, trong đó toàn bộ ứng dụng được xây dựng và triển khai trong một đơn vị duy nhất. Kiến trúc này cho phép dễ dàng quản lý và triển khai hệ thống, đặc biệt là trong các môi trường vừa và nhỏ.

**3. Công Nghệ và Kĩ Thuật**

* **Framework Django**: Django được chọn làm nền tảng phát triển chính cho hosSYS. Được xây dựng trên ngôn ngữ lập trình Python, Django cung cấp các tính năng mạnh mẽ và linh hoạt cho việc phát triển ứng dụng web.
* **Cơ Sở Dữ Liệu PostgreSQL**: PostgreSQL được sử dụng làm cơ sở dữ liệu chính cho hosSYS. PostgreSQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ và đáng tin cậy.
* **HTML/CSS/JavaScript**: Để phát triển giao diện người dùng, hosSYS sử dụng HTML, CSS và JavaScript, cùng với các thư viện và framework như Bootstrap để tạo ra giao diện thân thiện và dễ sử dụng.

**4. Quy Trình Phát Triển**

hosSYS được phát triển theo phương pháp phát triển phần mềm Agile, trong đó các vòng lặp ngắn và linh hoạt được sử dụng để liên tục phát triển, kiểm tra và cải thiện hệ thống.

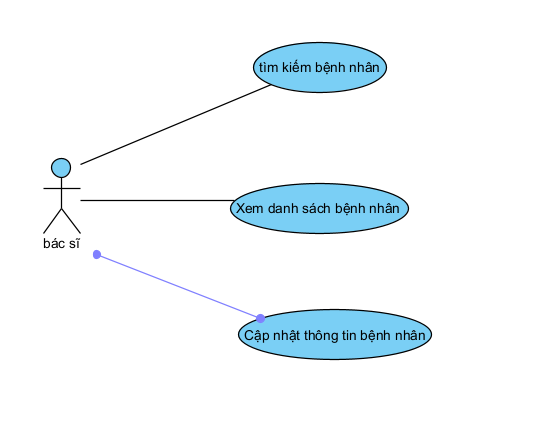
**5. Kiểm Thử và Triển Khai**

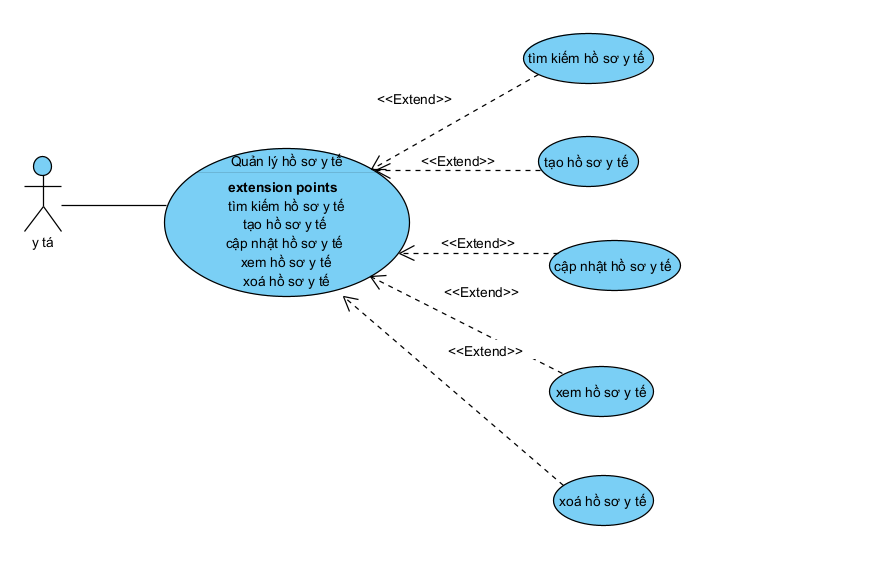
Trước khi triển khai, hosSYS đã trải qua quá trình kiểm thử chất lượng kỹ lưỡng, bao gồm kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống. Sau đó, hệ thống được triển khai một cách cẩn thận để đảm bảo tính ổn định và hiệu suất.

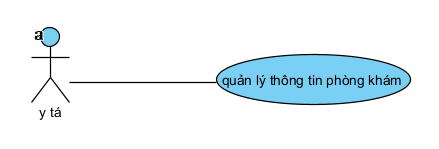
hosSYS là một hệ thống quản lý bệnh viện linh hoạt và mạnh mẽ, giúp cải thiện hiệu suất và chất lượng dịch vụ trong các cơ sở y tế.

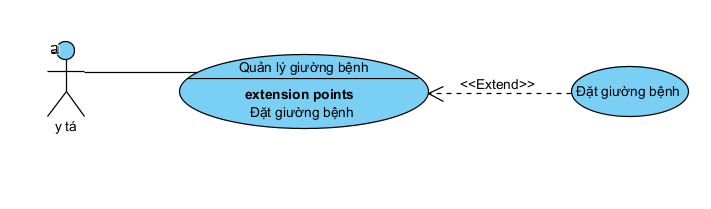
## 3.2 Các biểu đồ đặc tả hệ hosSys

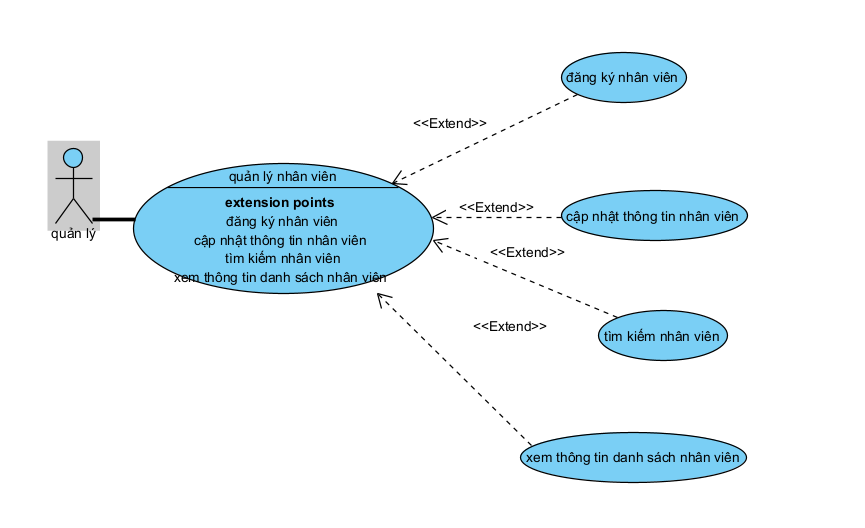
## 3.2.1 Biểu đồ use case

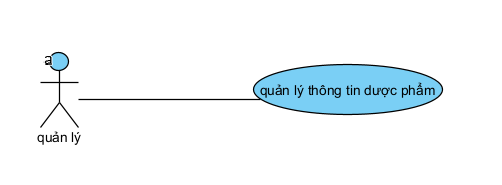


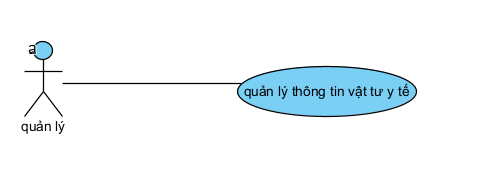


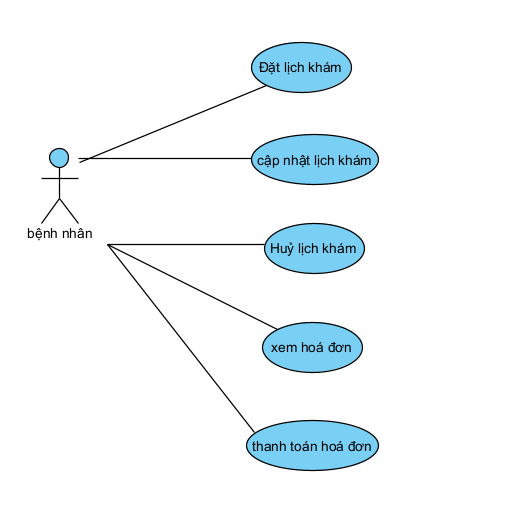


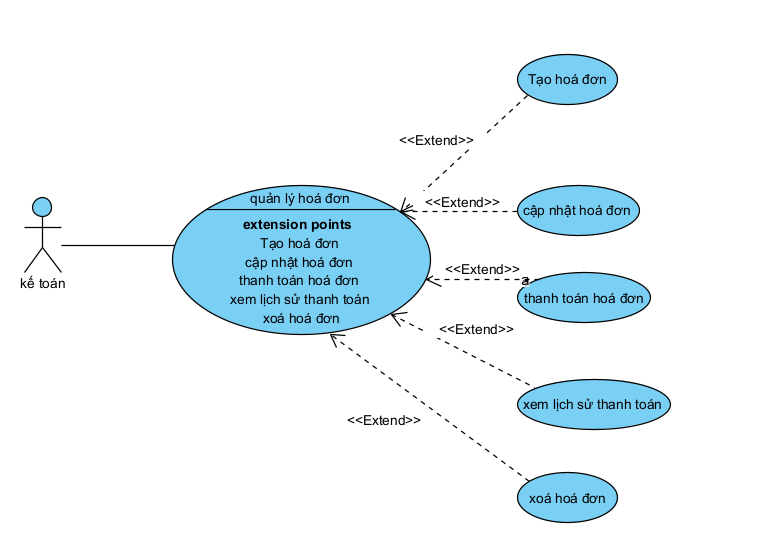




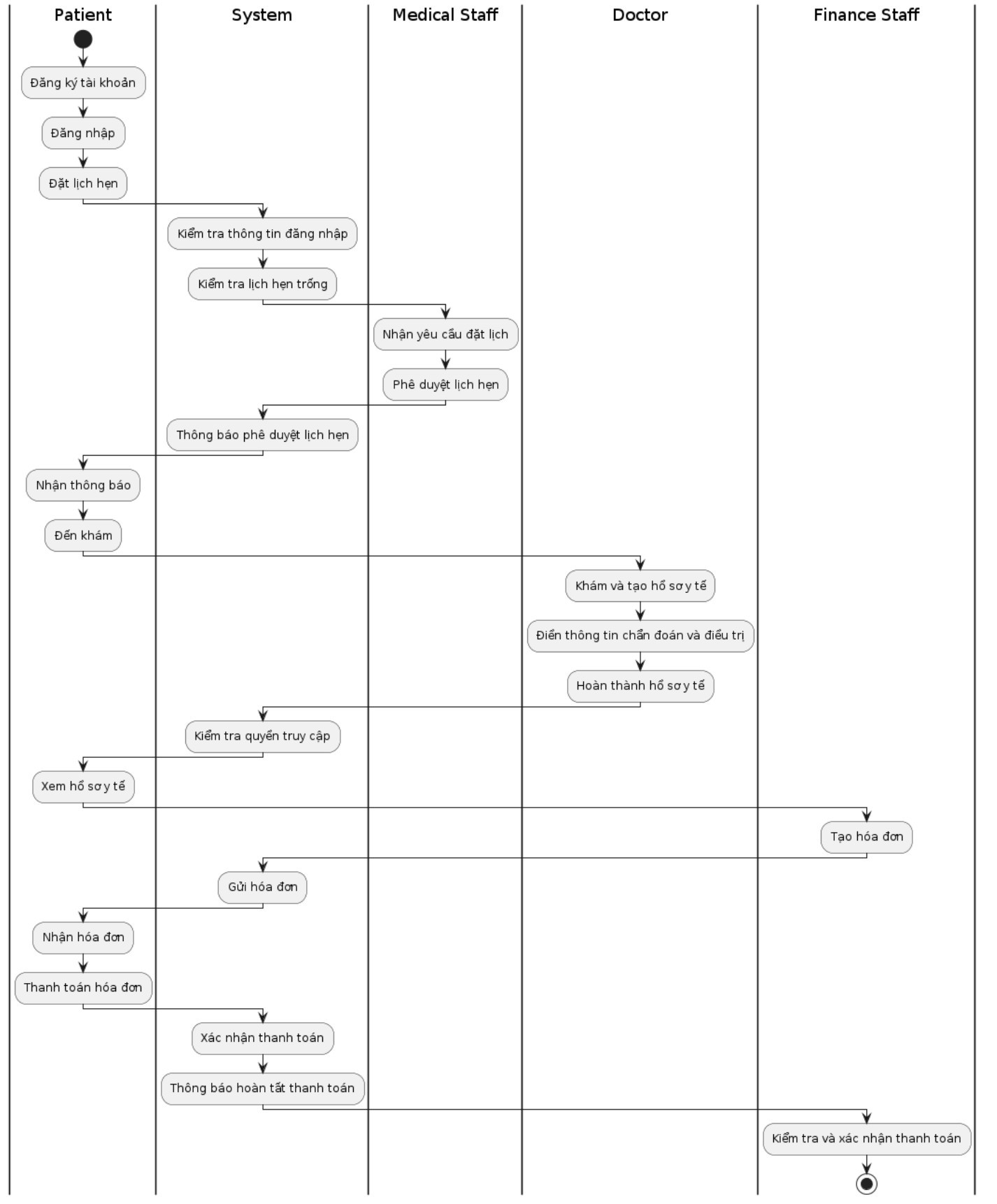








## 3.2.2 Biểu đồ hoạt động/swimlane

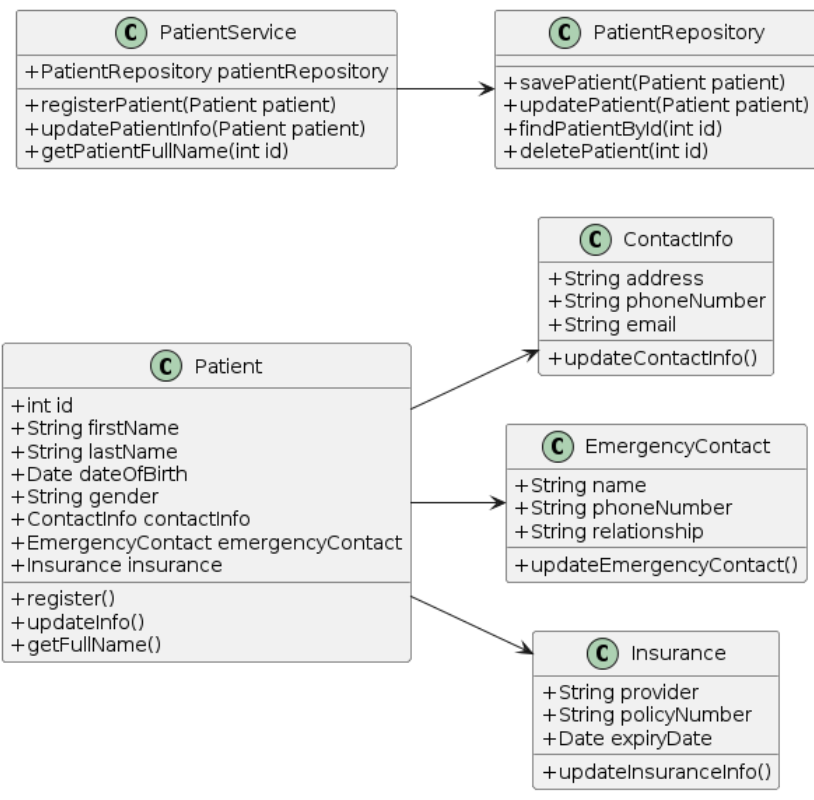


## 3.2.3 Biểu đồ lớp cho từng service

1. Quản lý Bệnh nhân (Patient Management)

Lớp và Mối quan hệ:

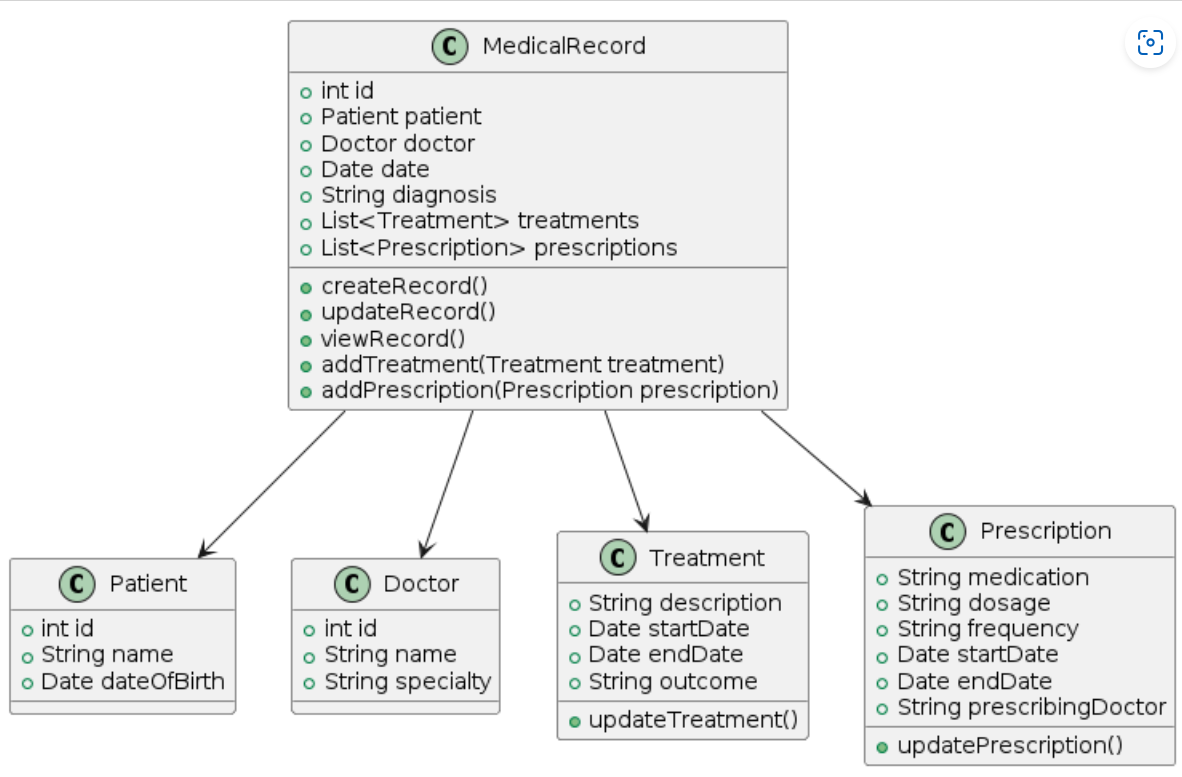
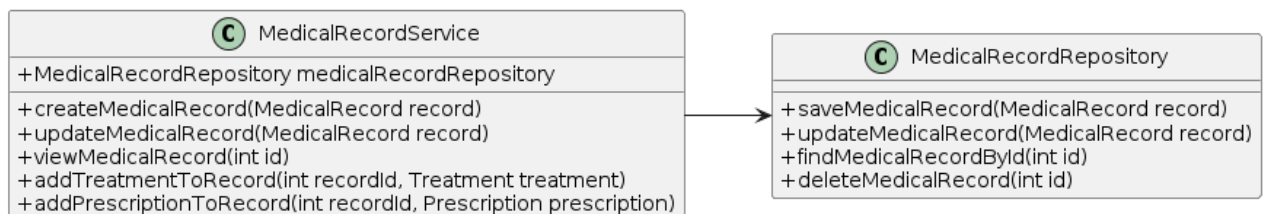
* Patient: Thông tin chi tiết về bệnh nhân.
* ContactInfo: Thông tin liên lạc của bệnh nhân.
* EmergencyContact: Thông tin liên hệ khẩn cấp của bệnh nhân.
* Insurance: Thông tin bảo hiểm y tế của bệnh nhân.

****

2. Quản lý Hồ sơ Y tế (Medical Records Management)

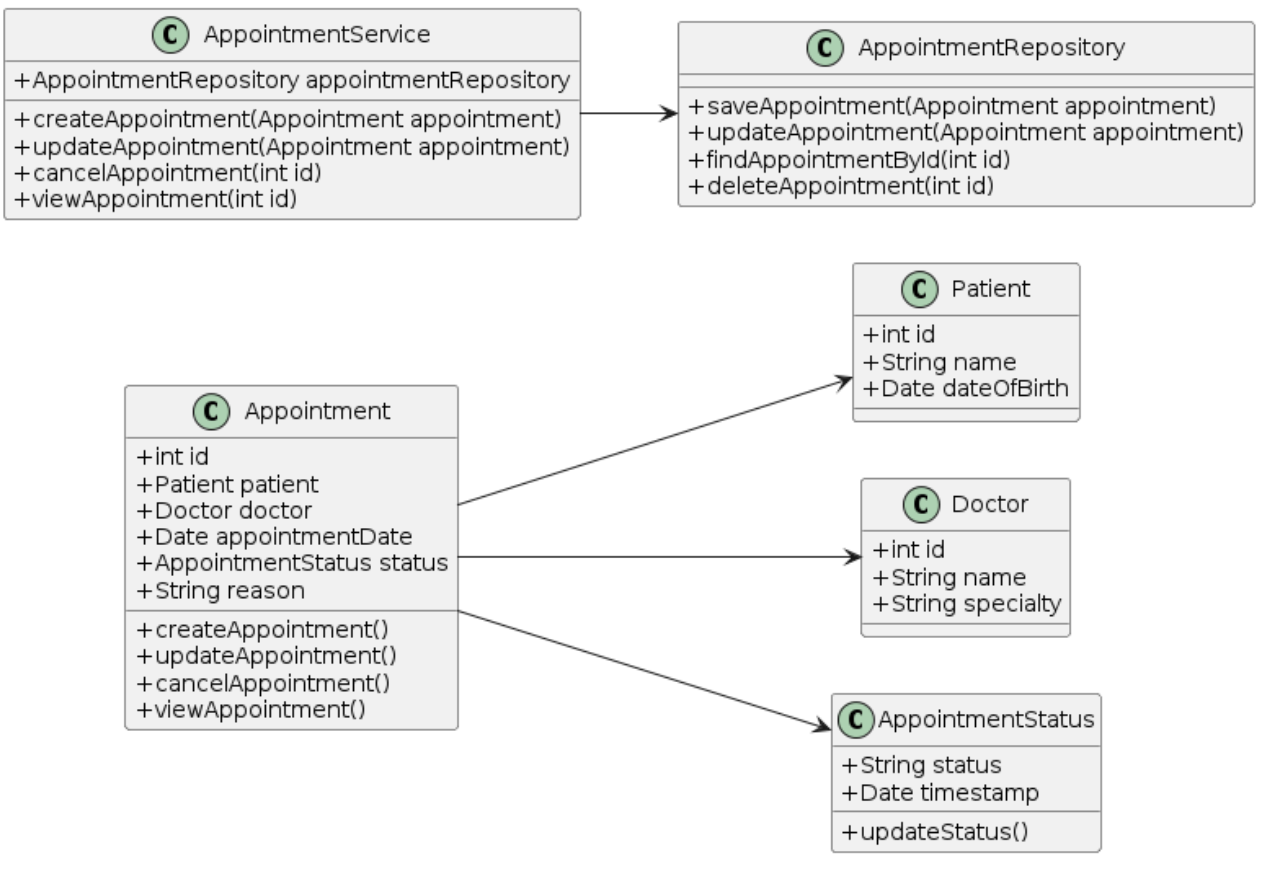
Lớp và Mối quan hệ:

* MedicalRecord: Hồ sơ y tế của bệnh nhân.
* Patient: Bệnh nhân.
* Doctor: Bác sĩ điều trị.
* Treatment: Chi tiết về các phương pháp điều trị.
* Prescription: Đơn thuốc được kê cho bệnh nhân.

****3. Quản lý Lịch hẹn (Appointment Management)

Lớp và Mối quan hệ:

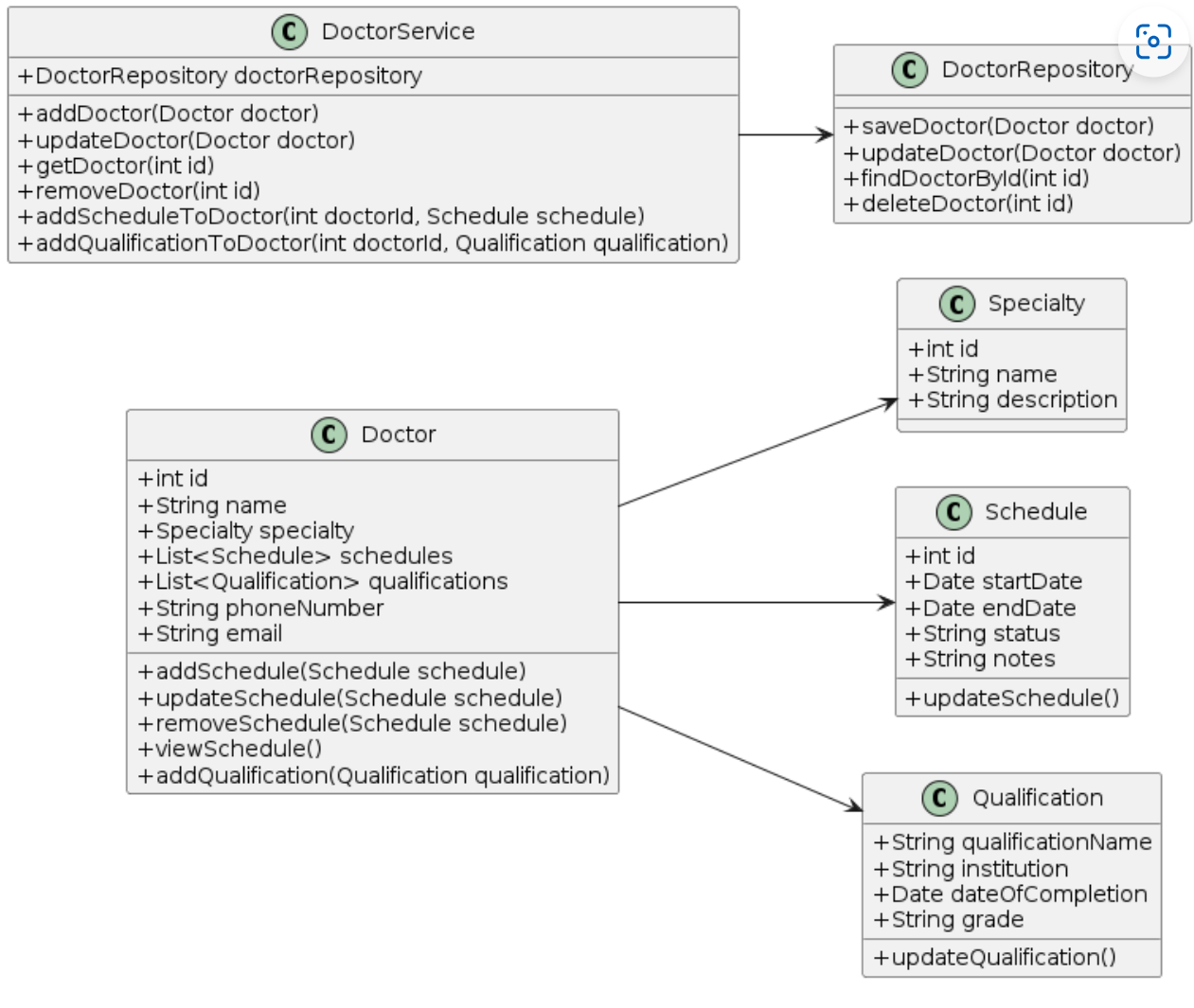
* Appointment: Lịch hẹn.
* Patient: Bệnh nhân đặt lịch.
* Doctor: Bác sĩ thực hiện lịch hẹn.

****

4. Quản lý Bác sĩ (Doctor Management)

Lớp và Mối quan hệ:

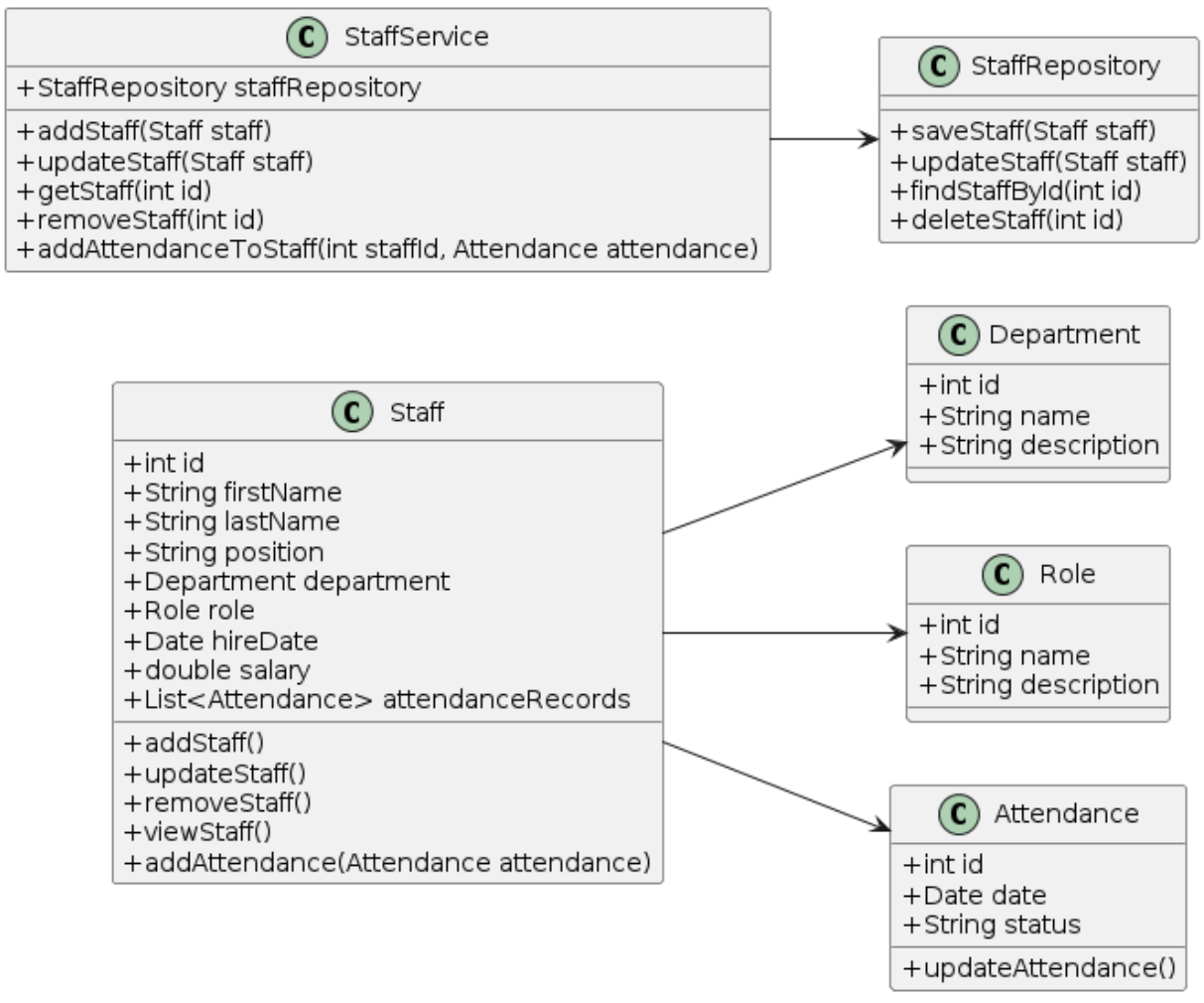
* Doctor: Thông tin về bác sĩ.
* Specialty: Chuyên khoa của bác sĩ.
* Schedule: Lịch làm việc của bác sĩ.



5. Quản lý Nhân viên (Staff Management)

Lớp và Mối quan hệ:

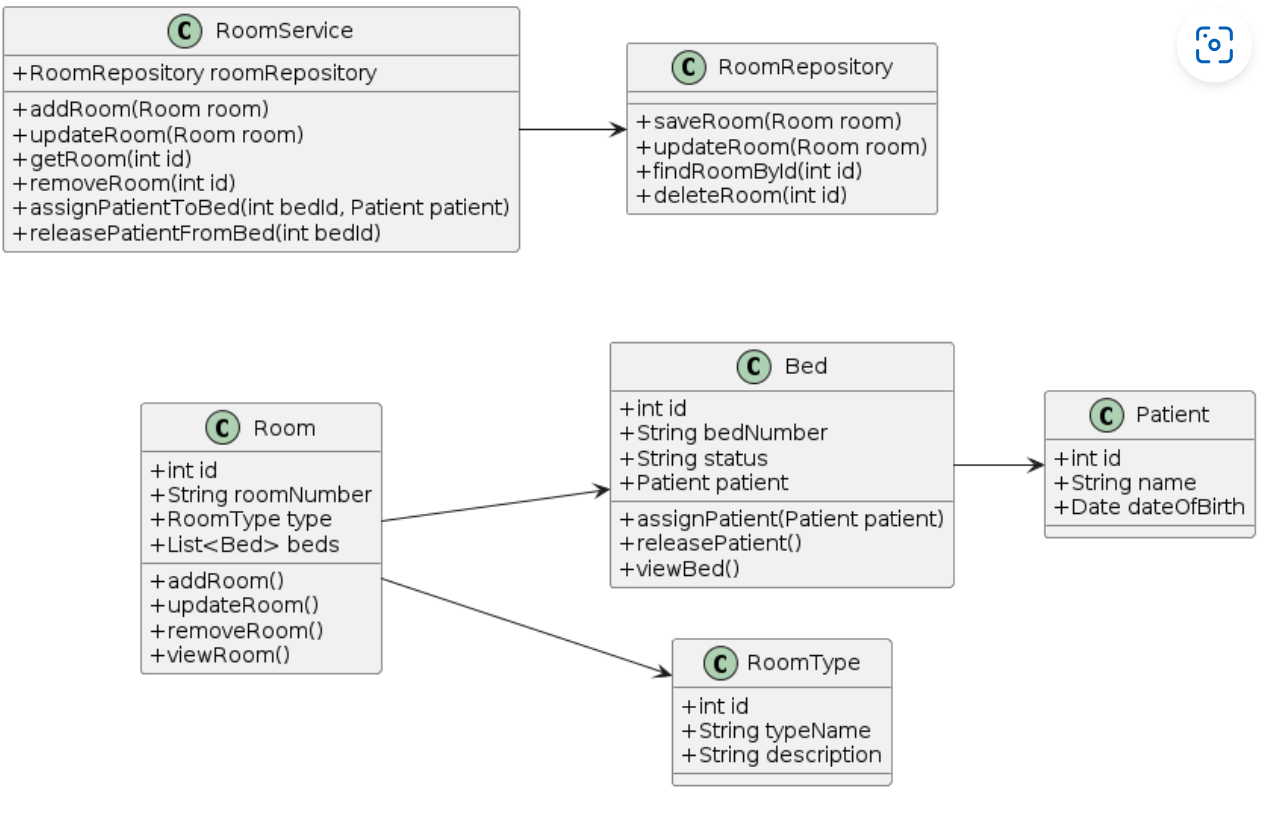
* Staff: Thông tin về nhân viên.
* Department: Bộ phận làm việc của nhân viên**.**

****

6. Quản lý Phòng khám và Giường bệnh (Room and Bed Management)

Lớp và Mối quan hệ:

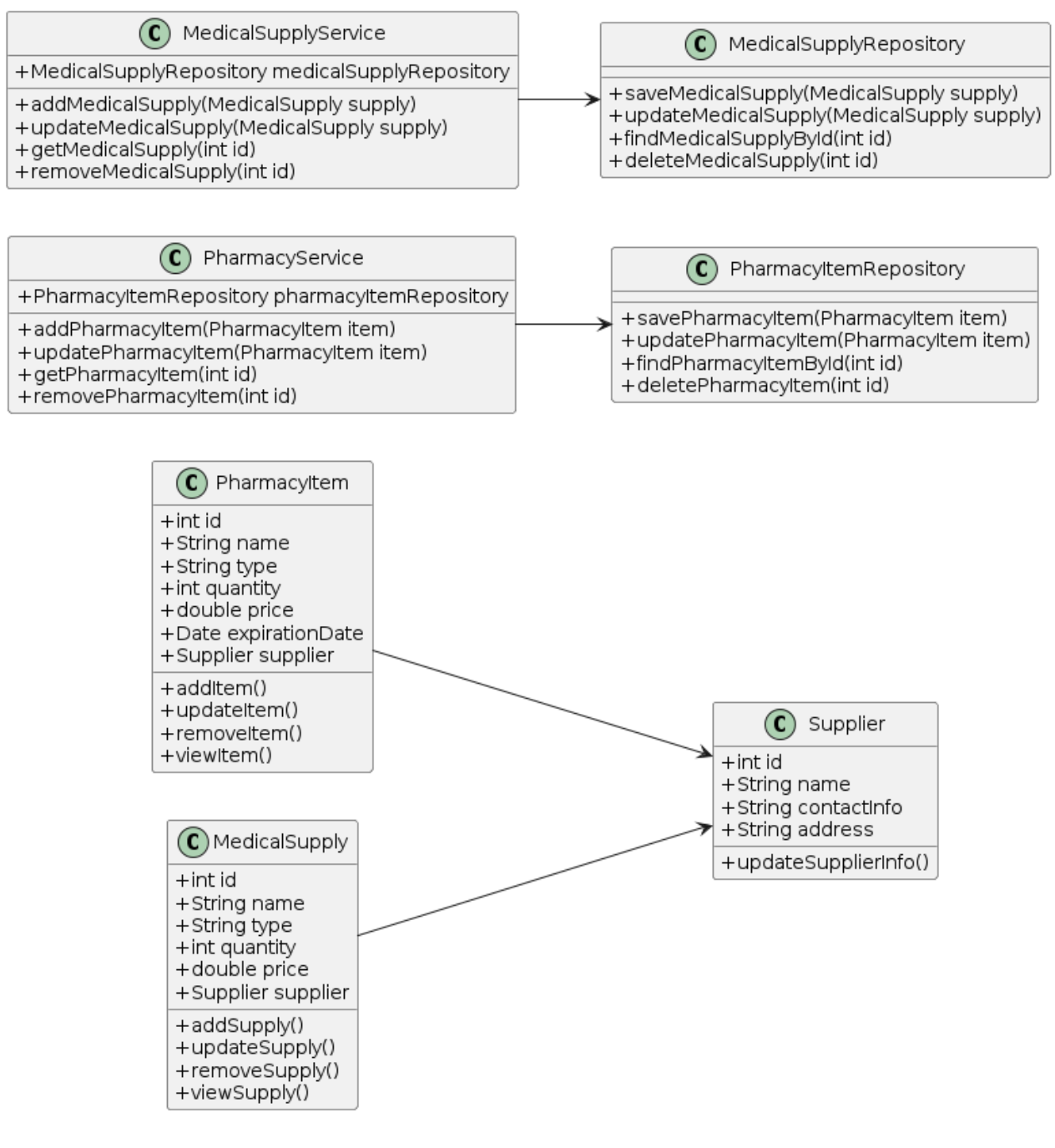
* Room: Thông tin về phòng khám.
* Bed: Thông tin về giường bệnh.
* Patient: Bệnh nhân sử dụng giường bệnh.

****

7. Quản lý Dược phẩm và Vật tư Y tế (Pharmacy and Medical Supplies Management)

Lớp và Mối quan hệ:

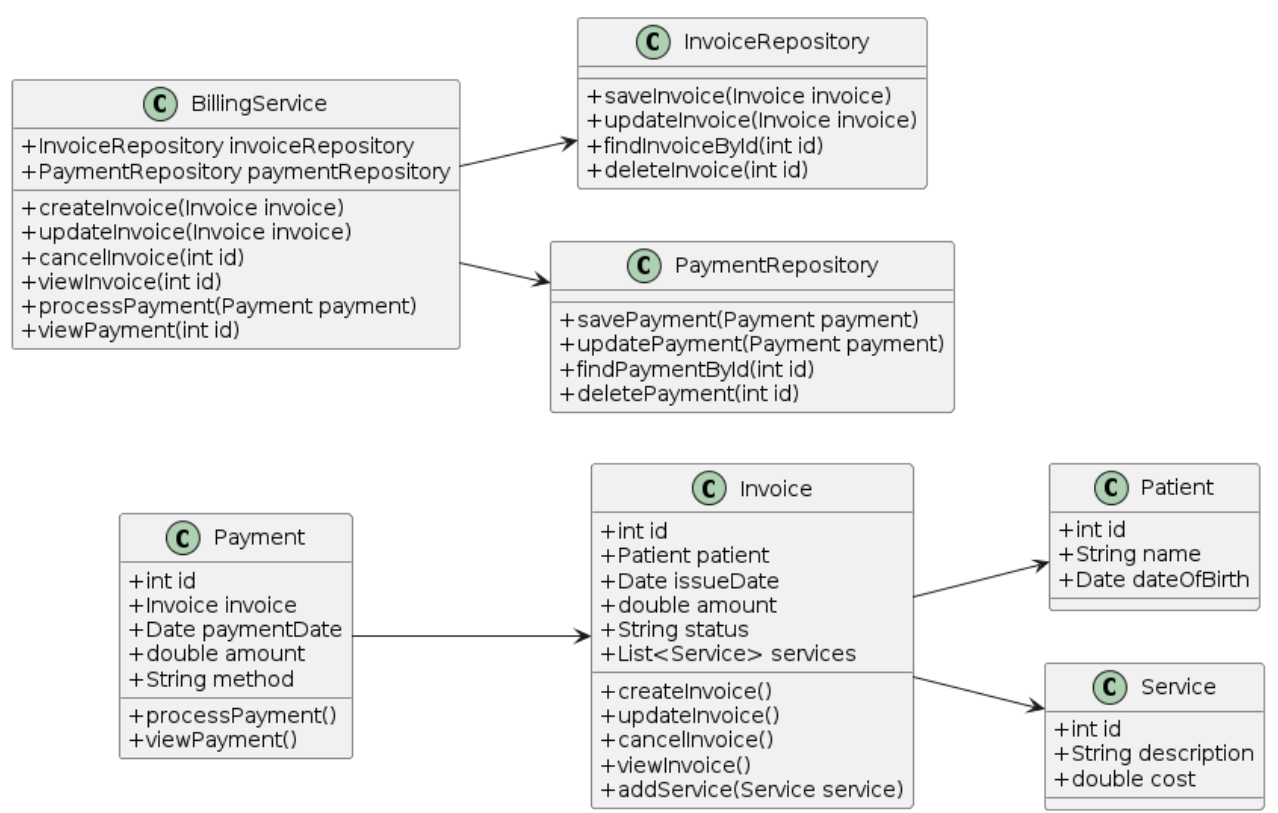
* PharmacyItem: Thông tin về dược phẩm.
* MedicalSupply: Thông tin về vật tư y tế.



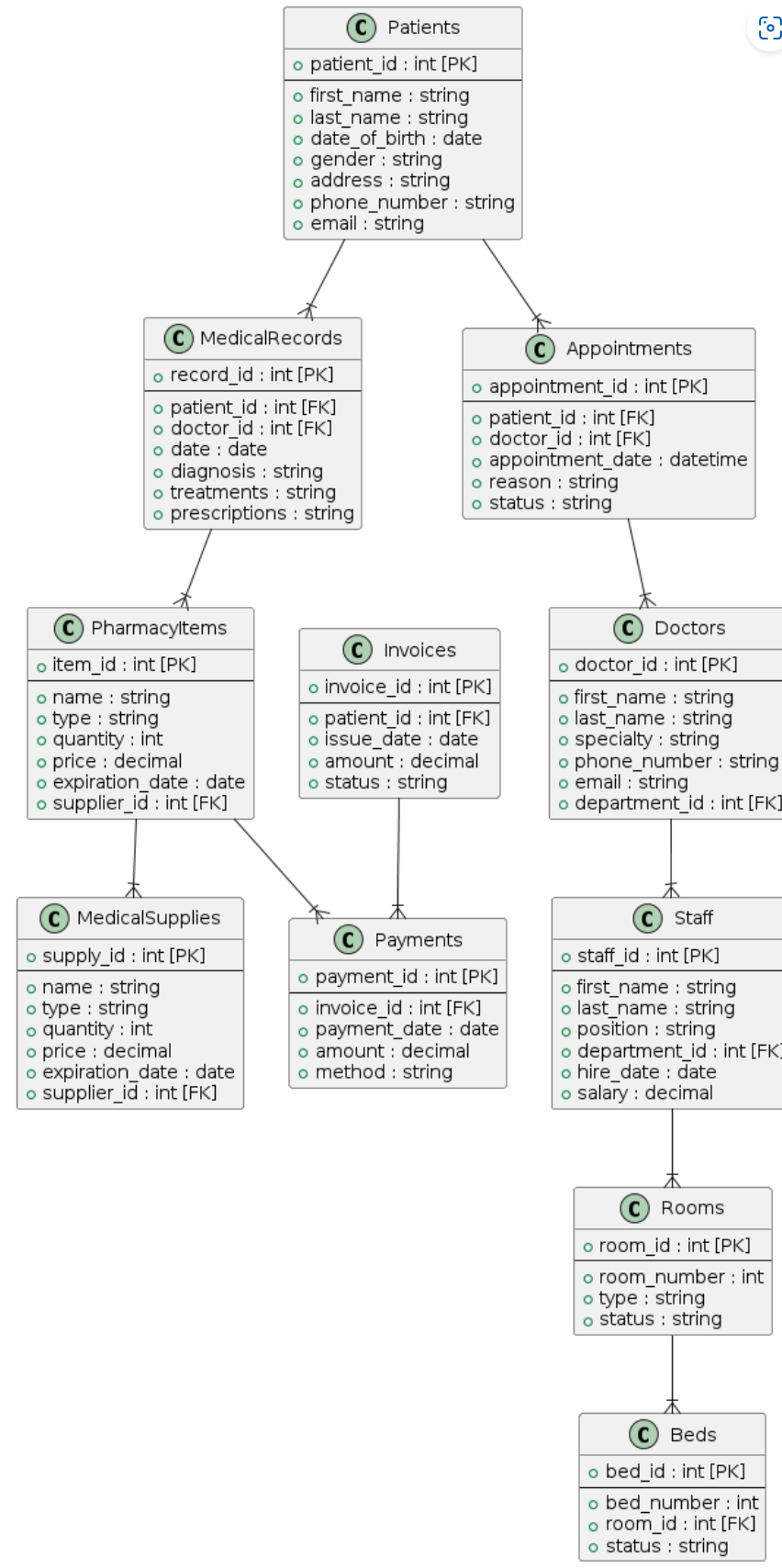
8. Quản lý Hóa đơn và Thanh toán (Billing and Payment Management)

Lớp và Mối quan hệ:

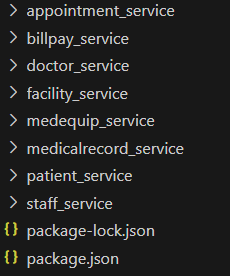
* Invoice: Hóa đơn.
* Payment: Thanh toán.
* Patient: Bệnh nhân liên quan đến hóa đơn.



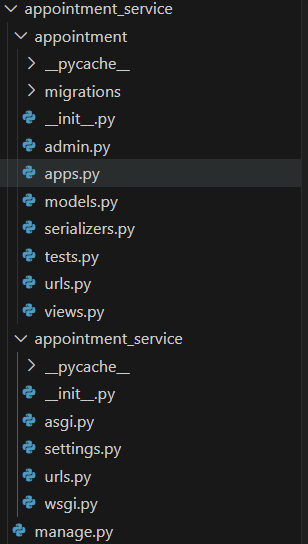
**Biểu đồ lớp cho hosSys:**

****

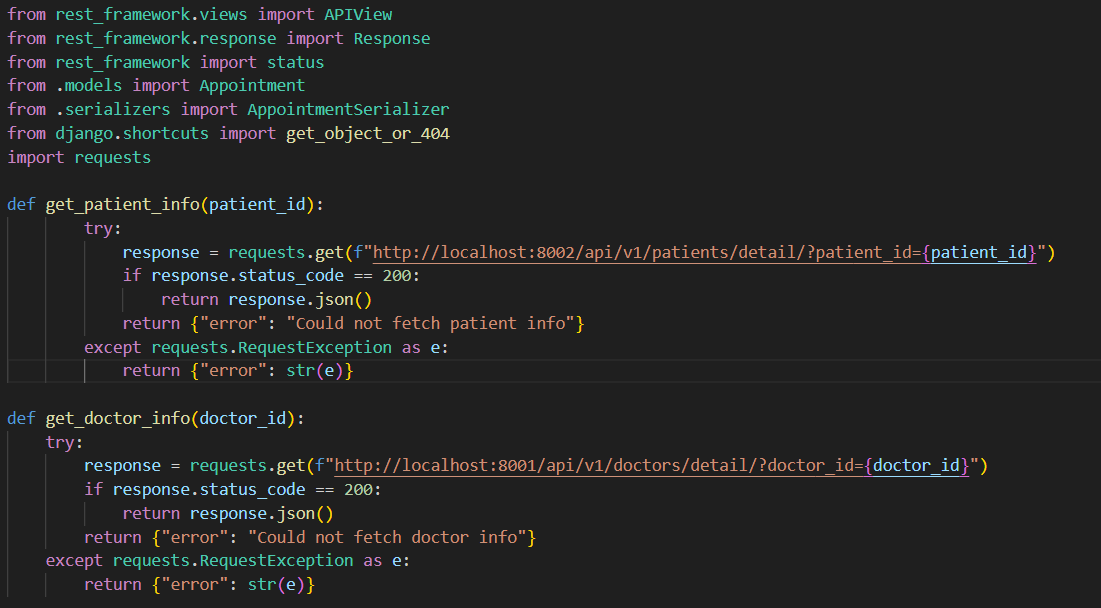
## 3.3 Xây dựng hệ hosSys

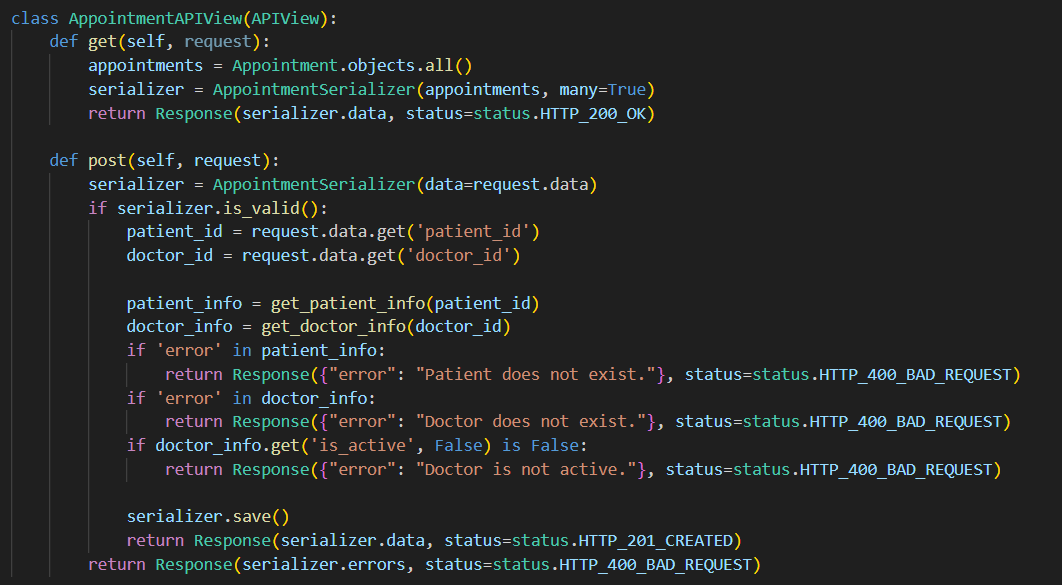


Appointment\_service:

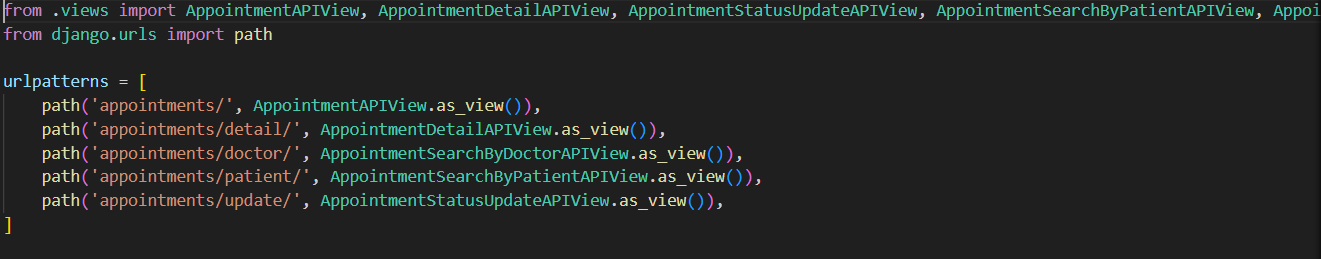




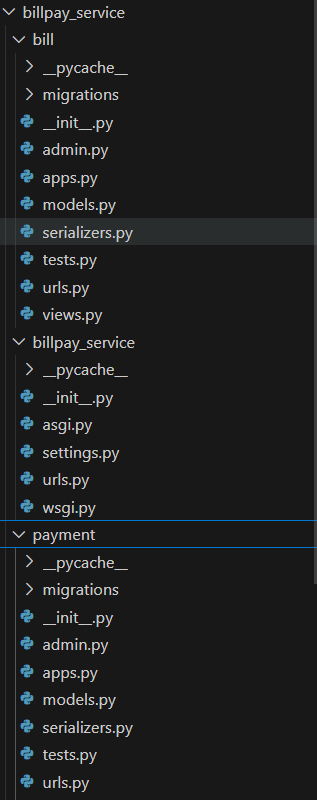


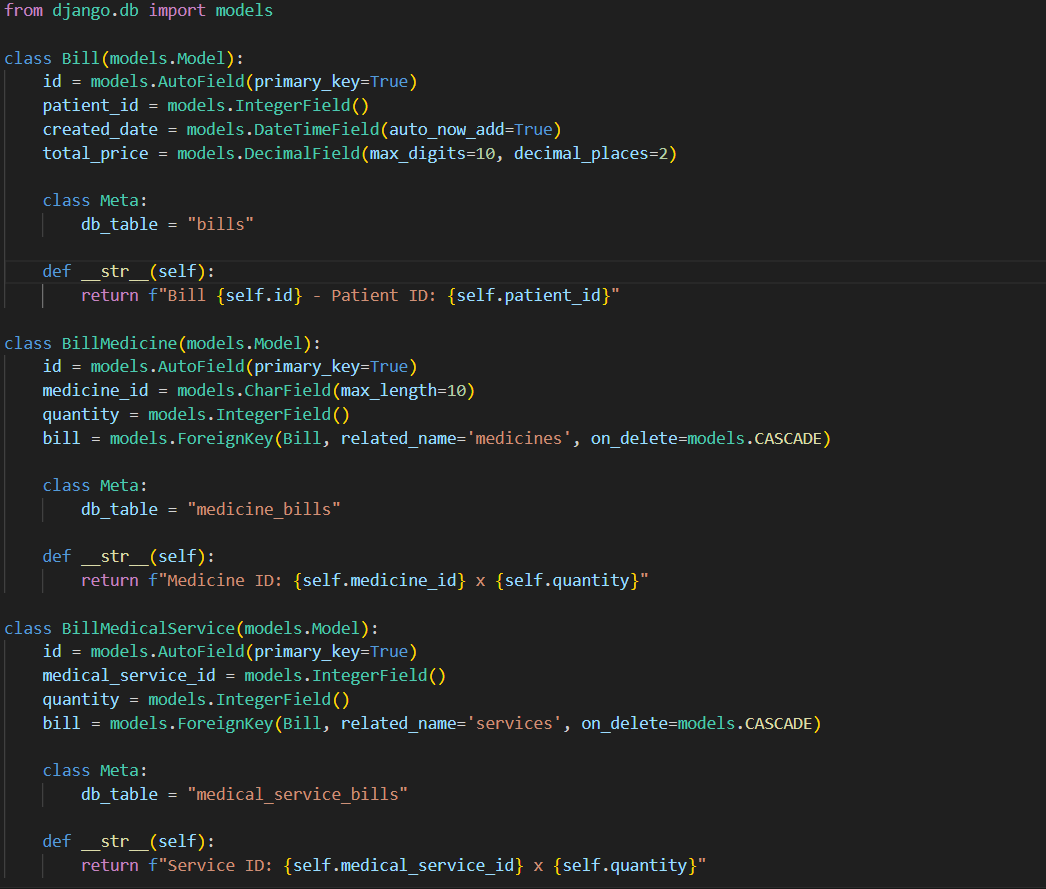


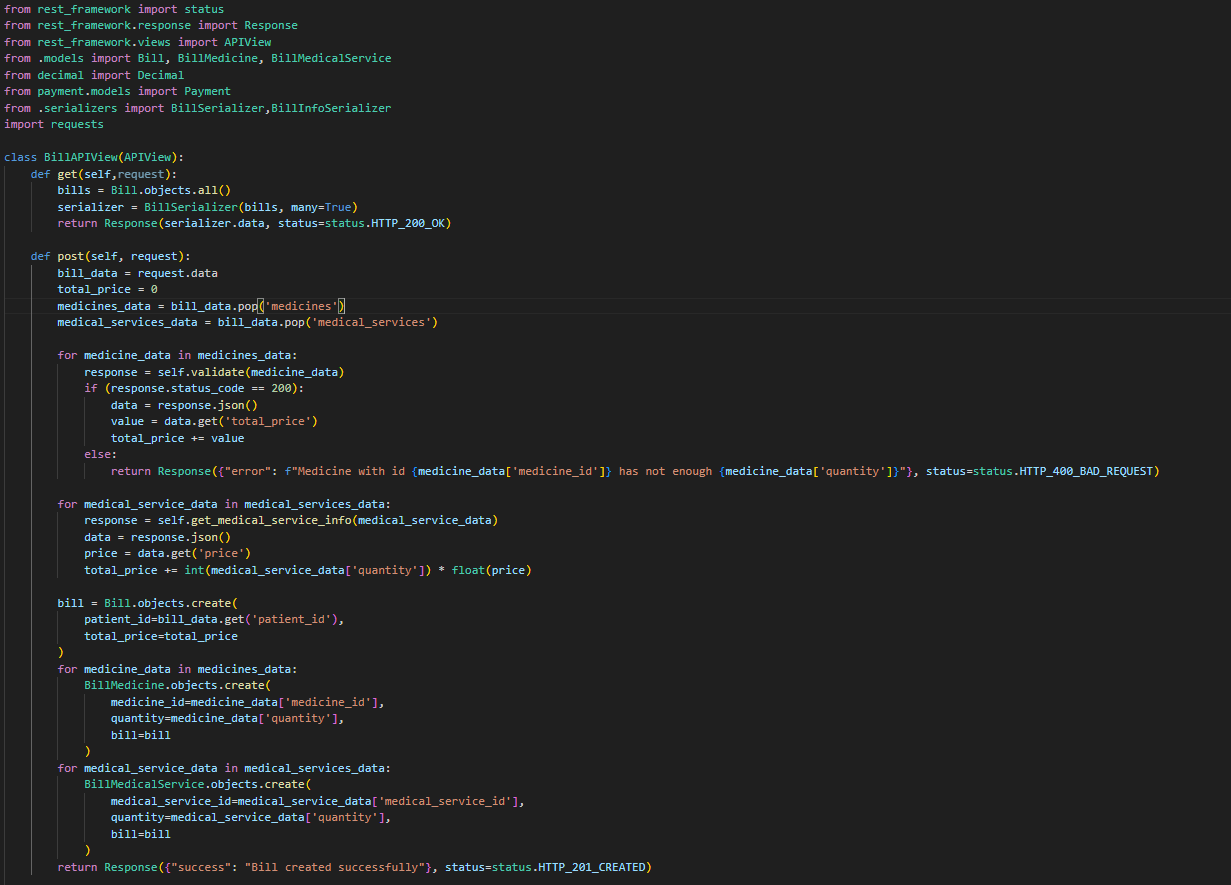


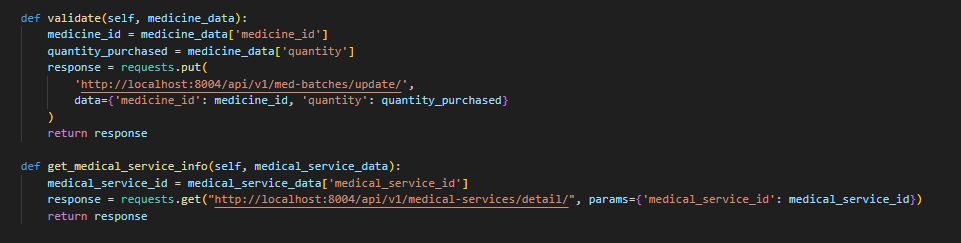


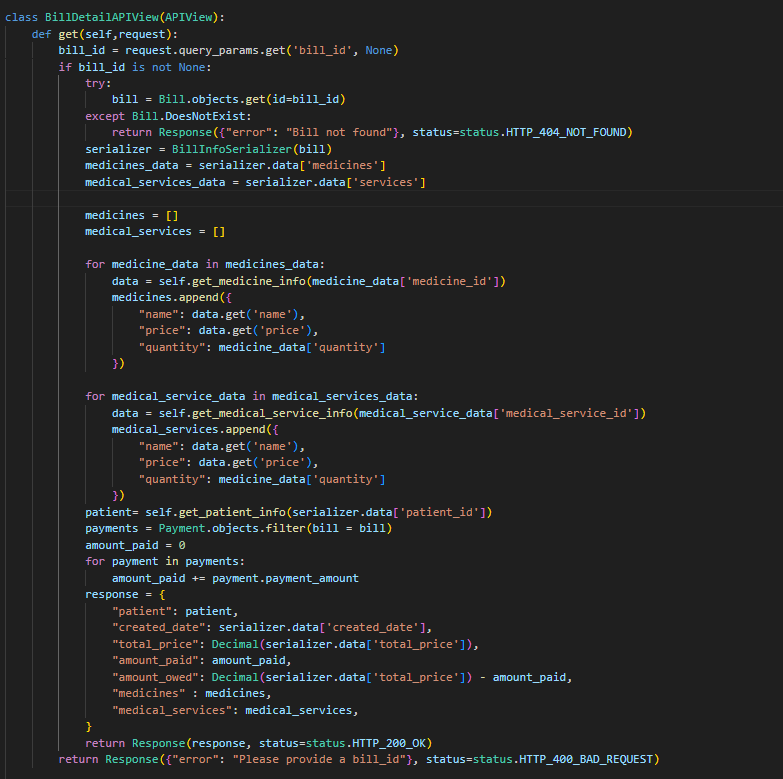
Billpay\_service:

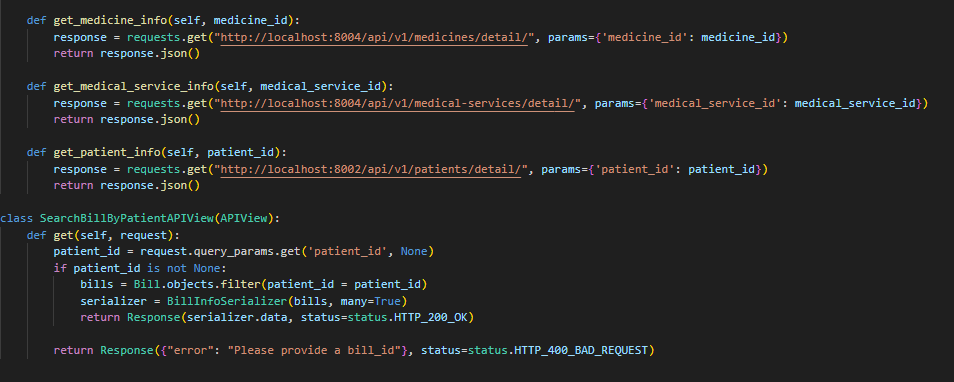


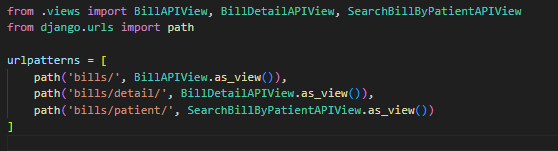


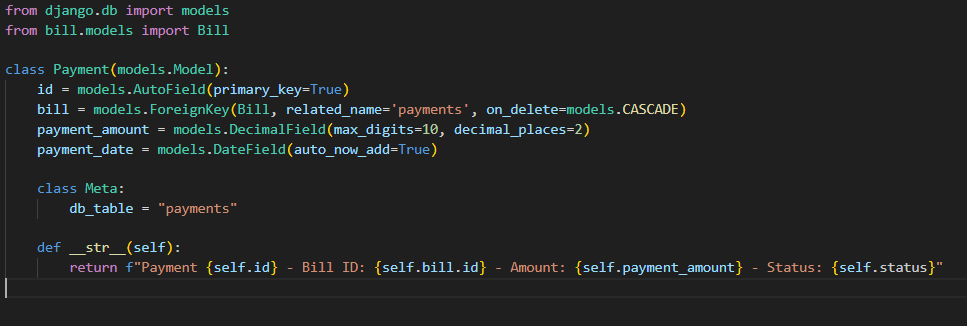


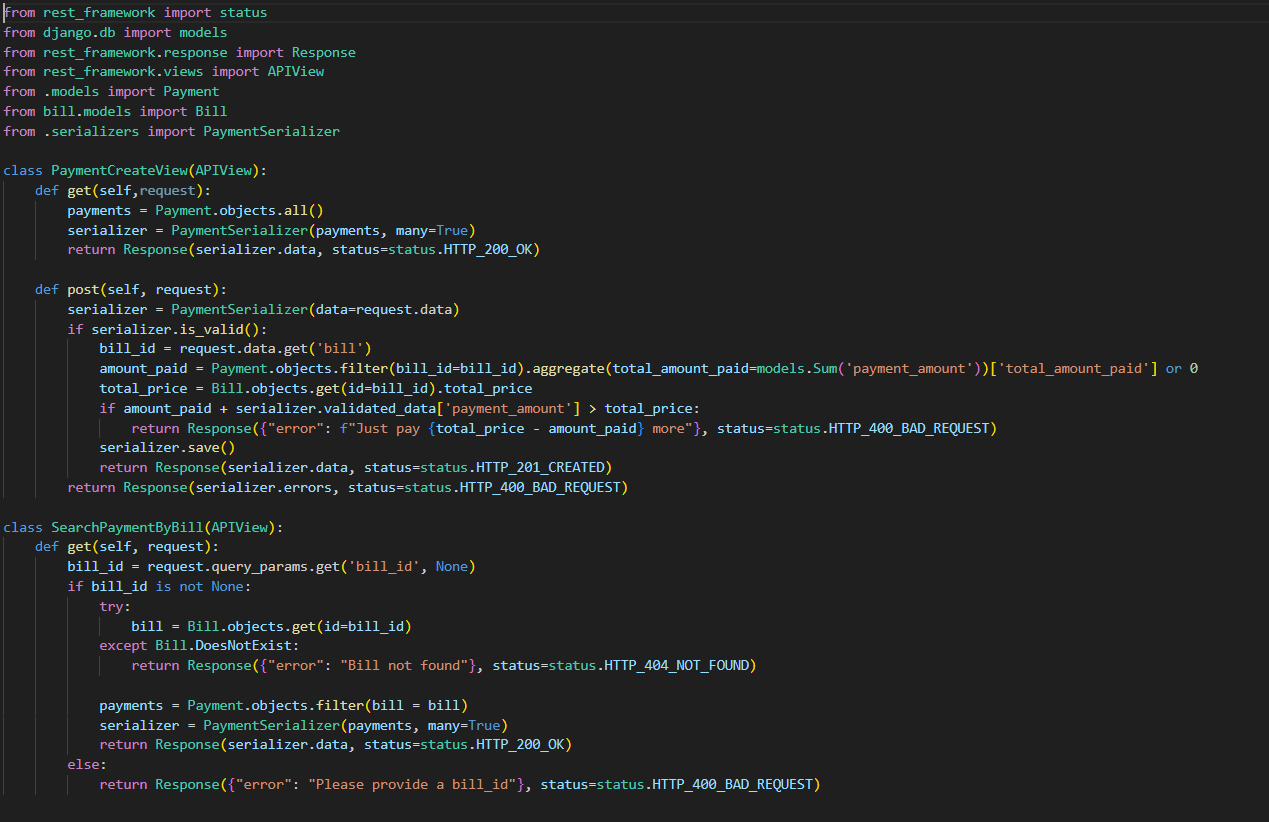


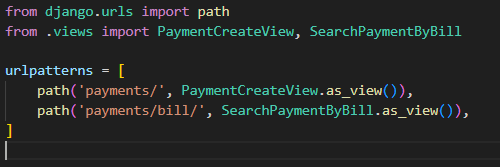




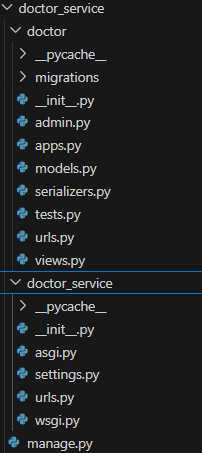


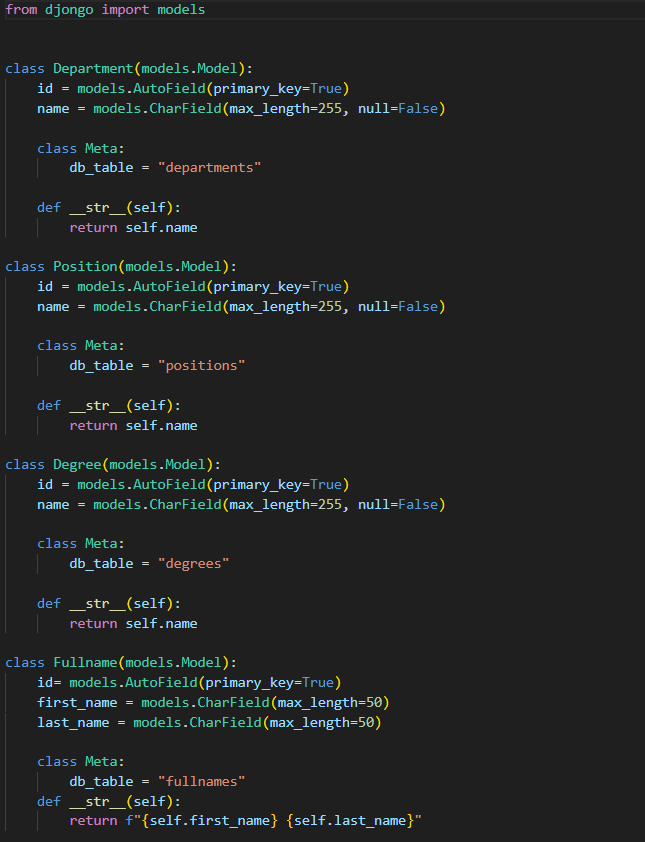


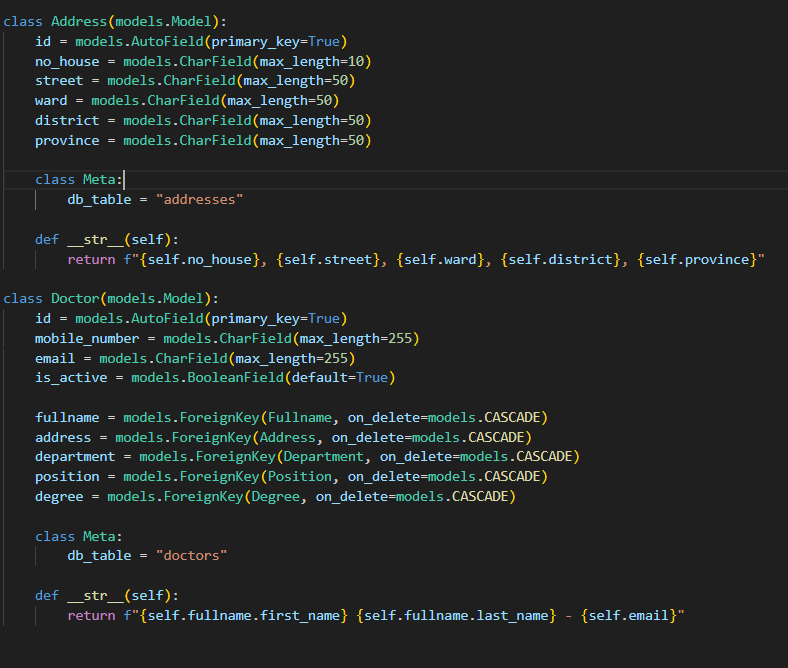


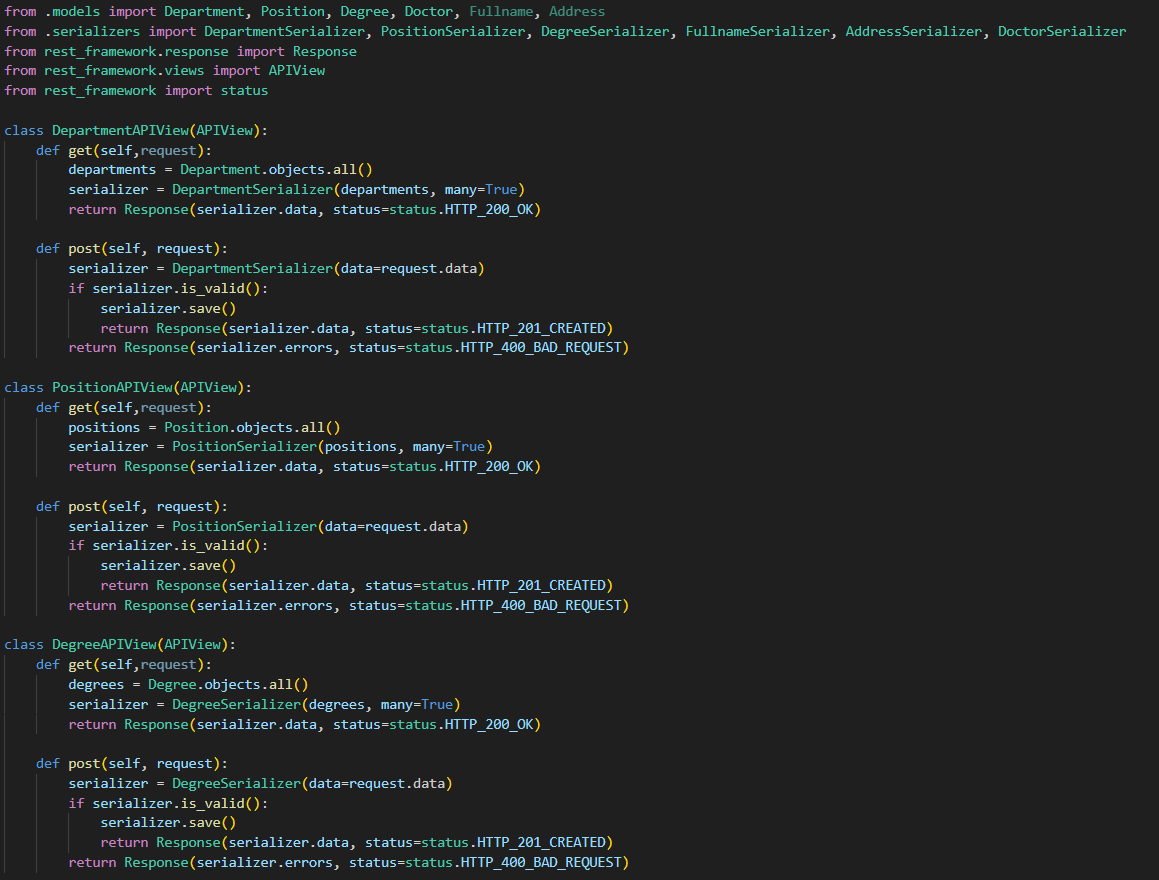


Doctor\_service:

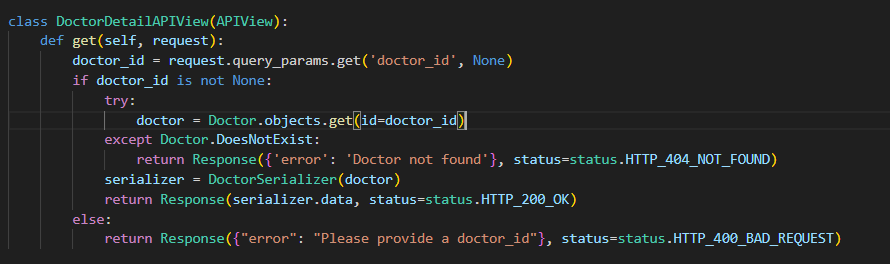


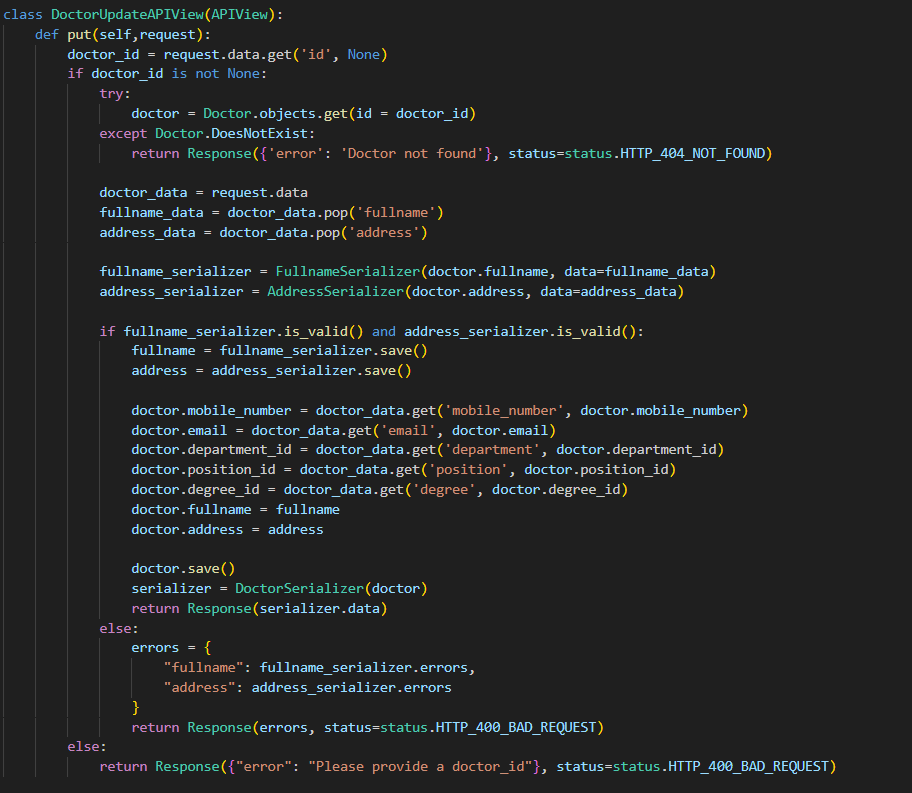


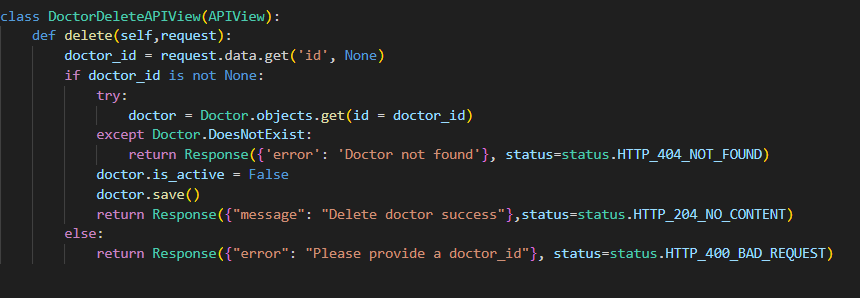


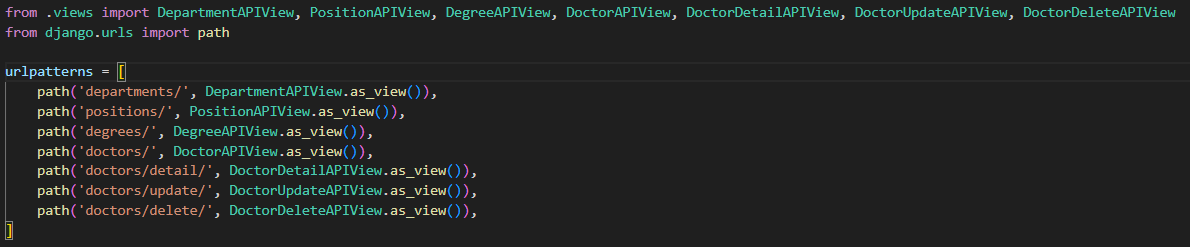




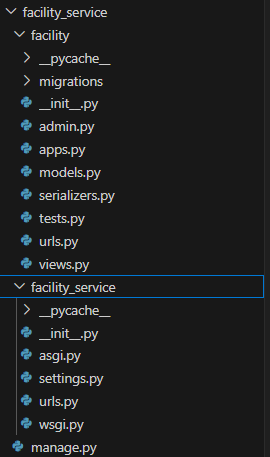


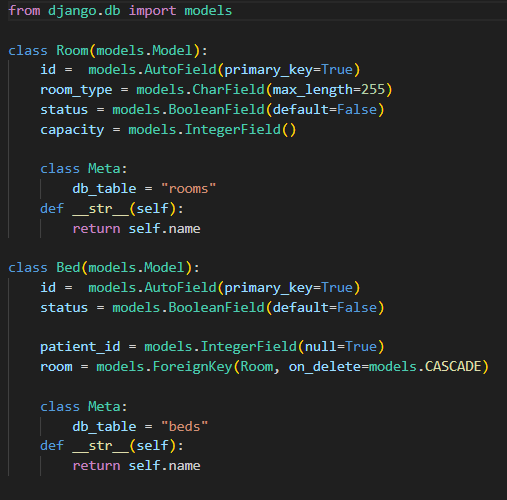


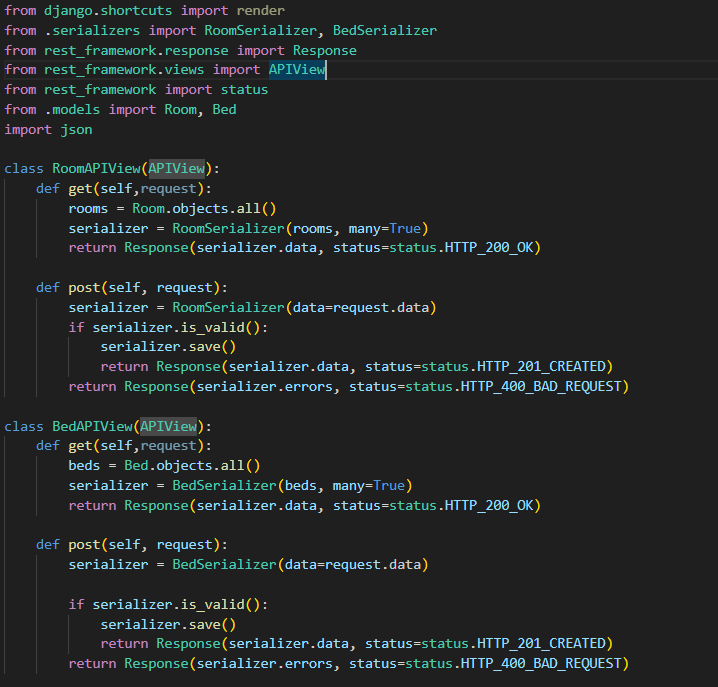


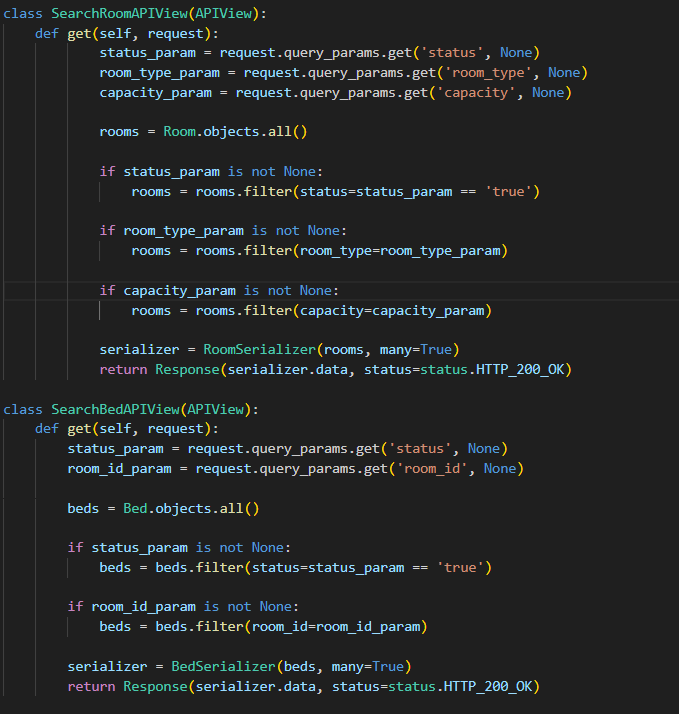


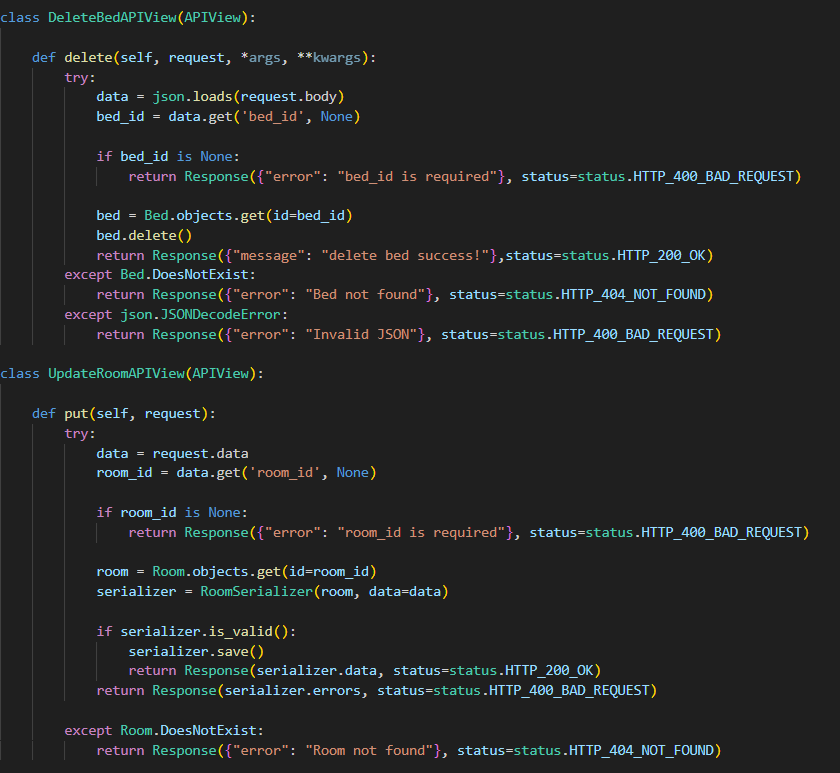
Facility\_service:

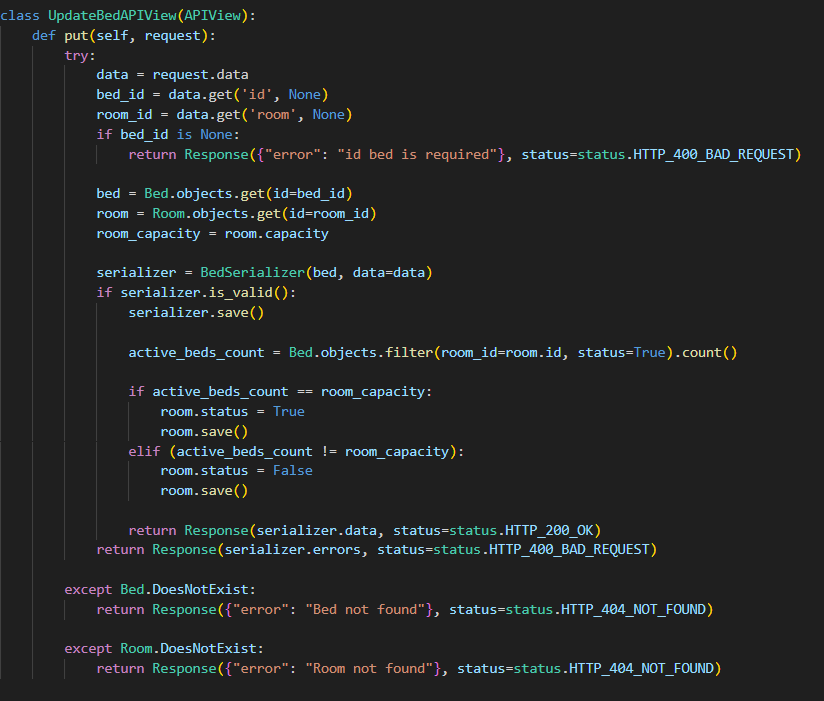


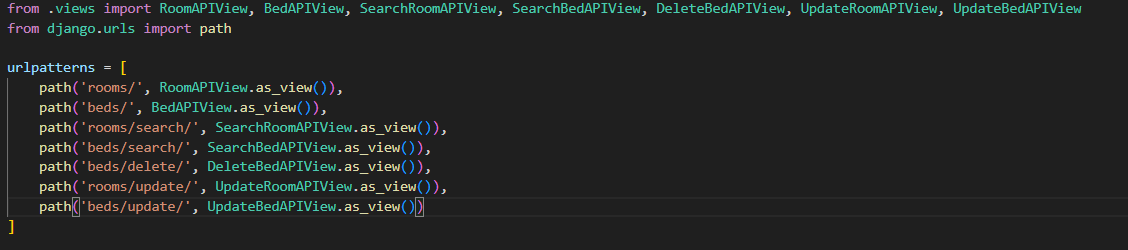




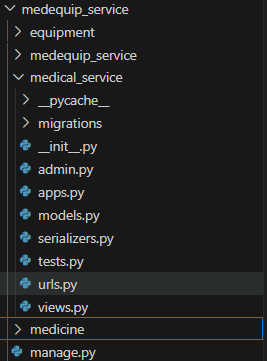


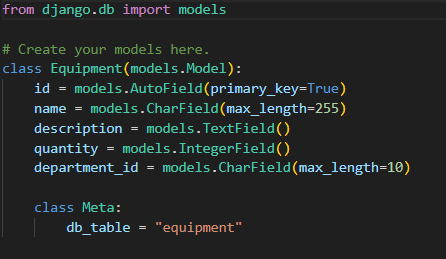


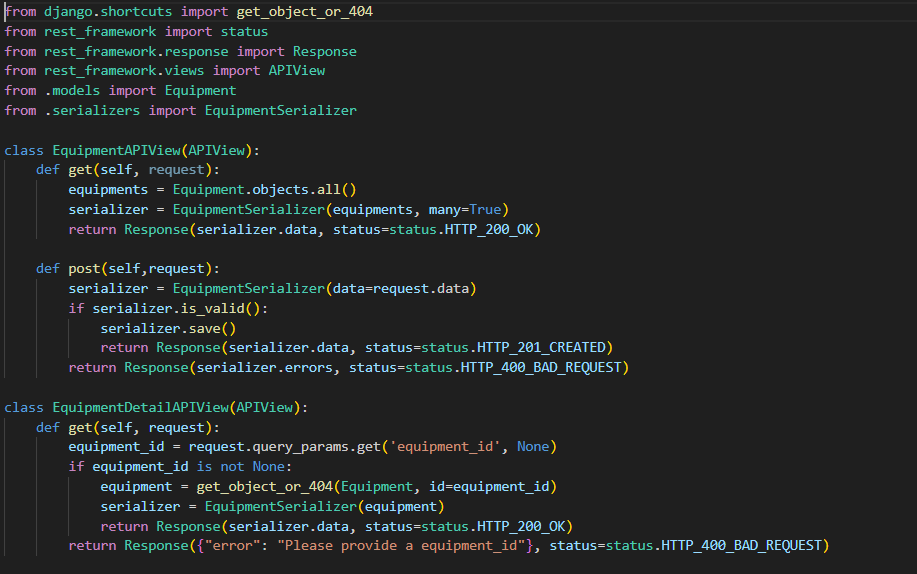


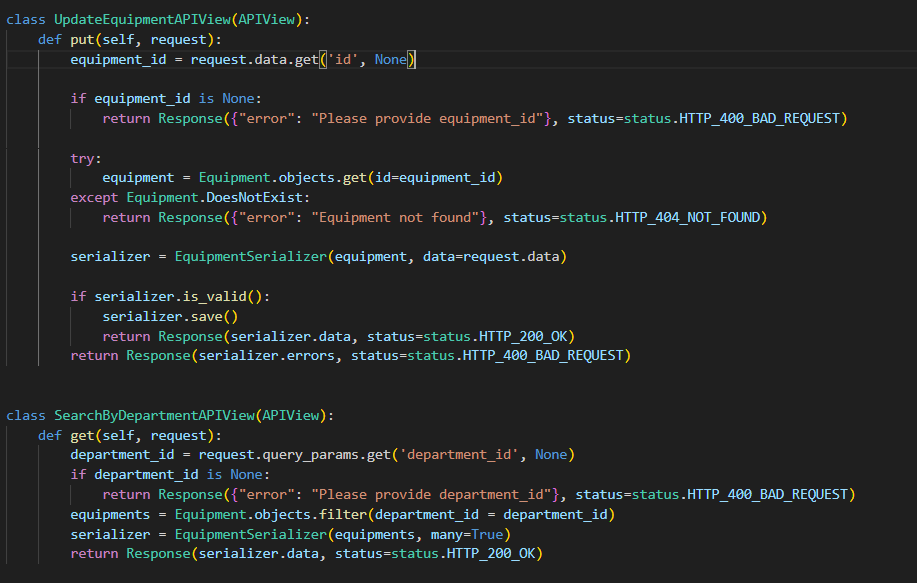


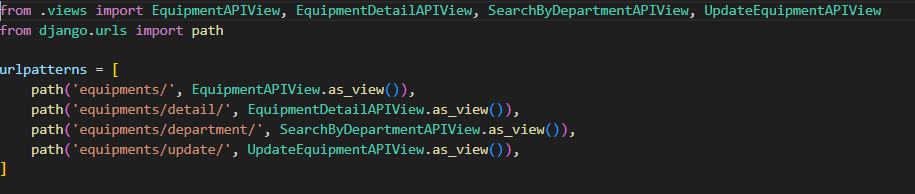
Medequip\_service:

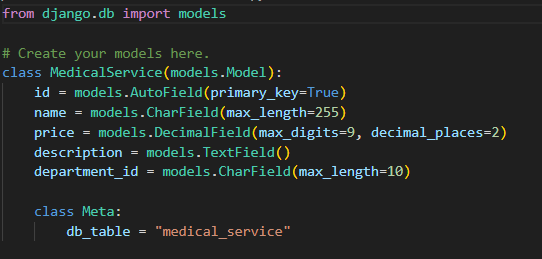


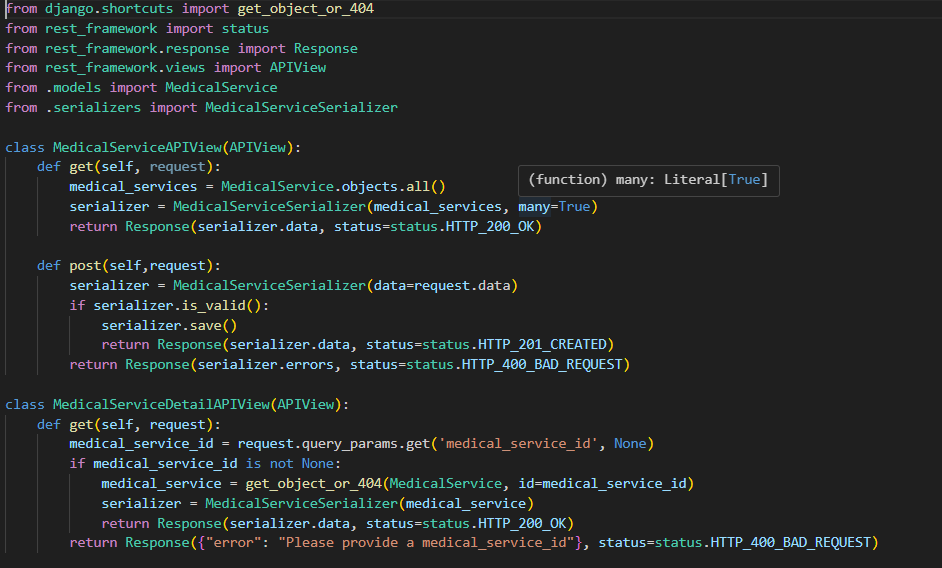


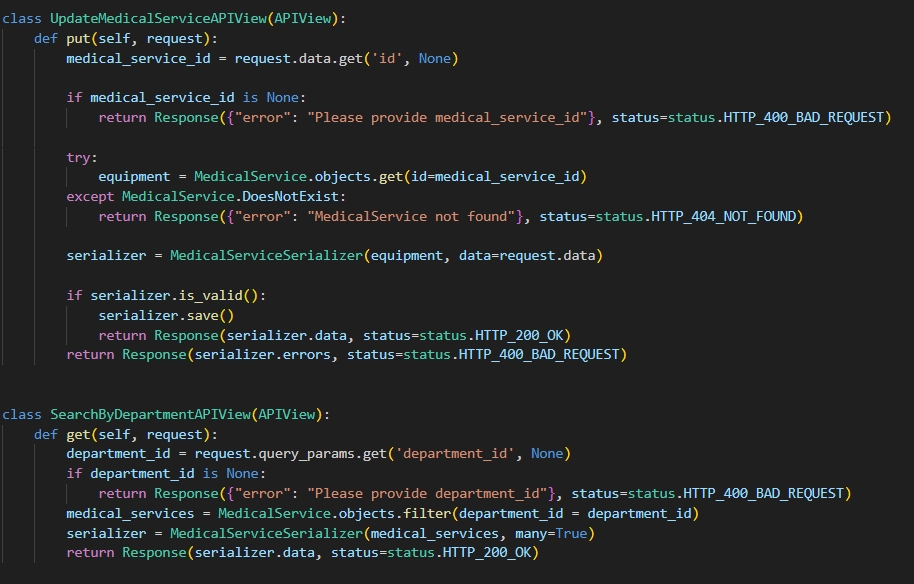


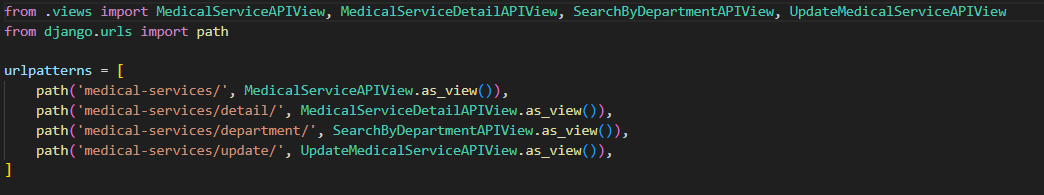


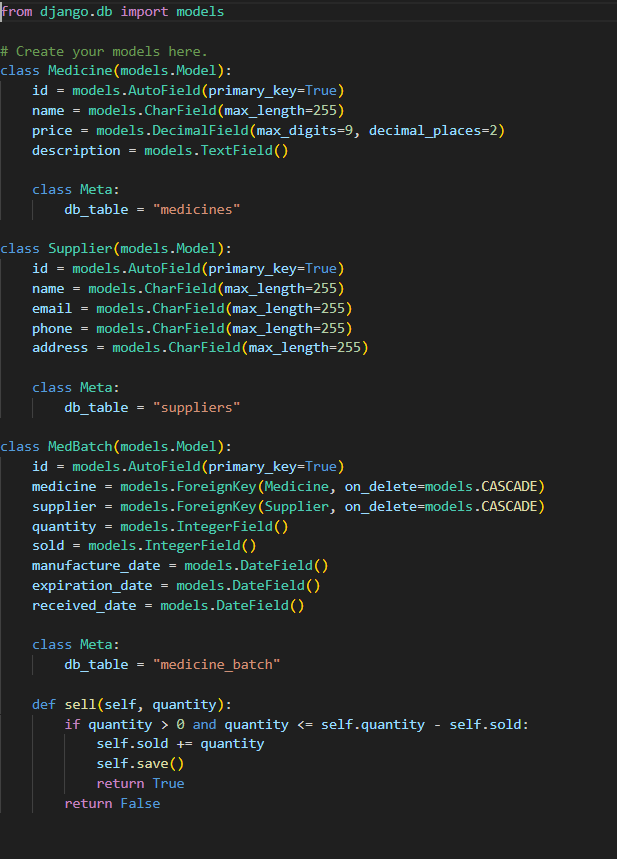


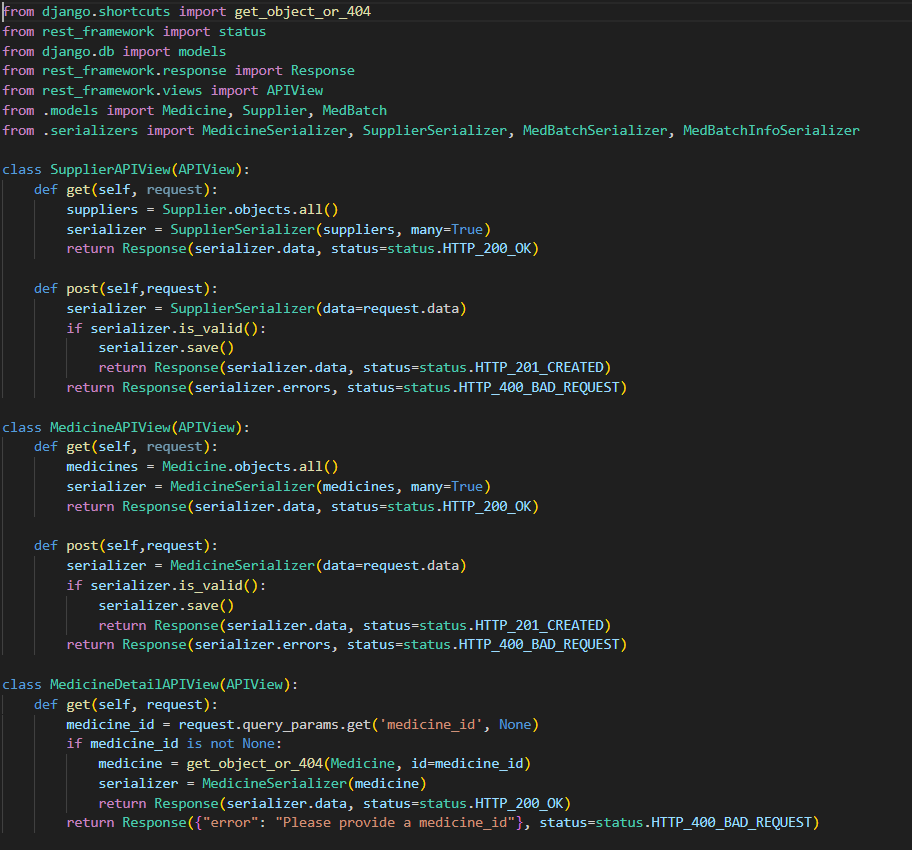


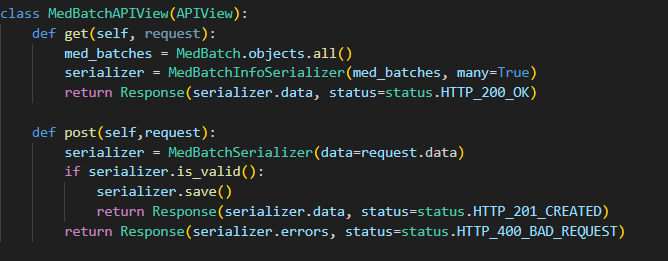


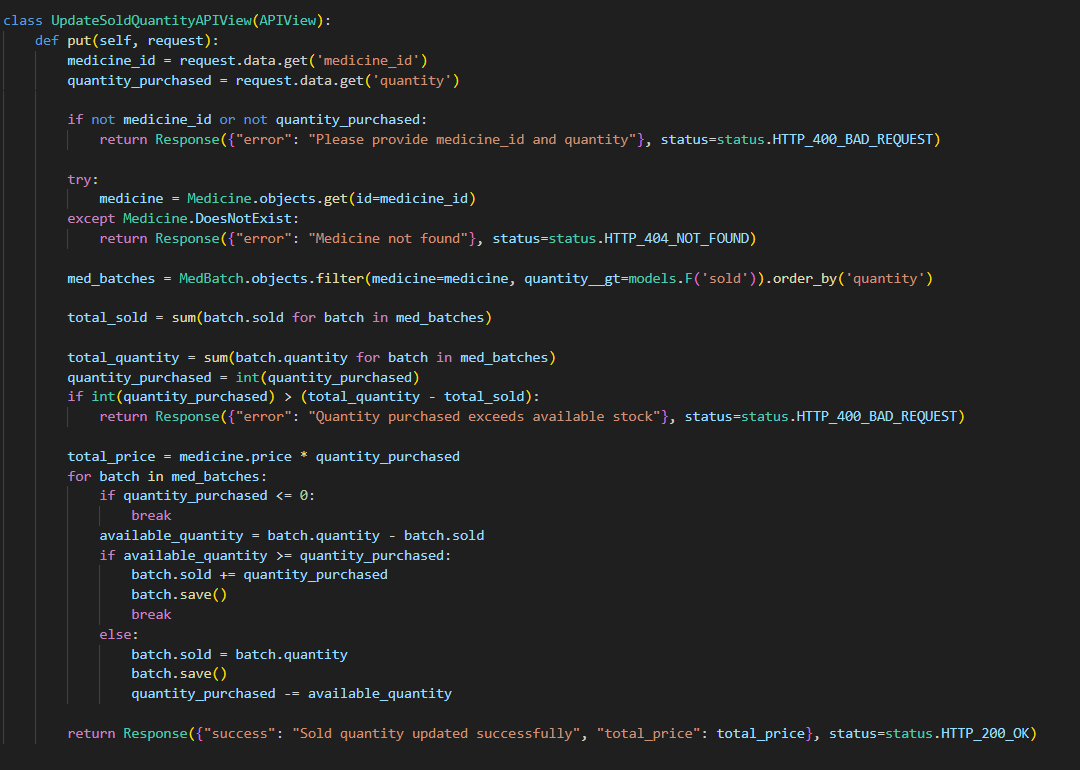


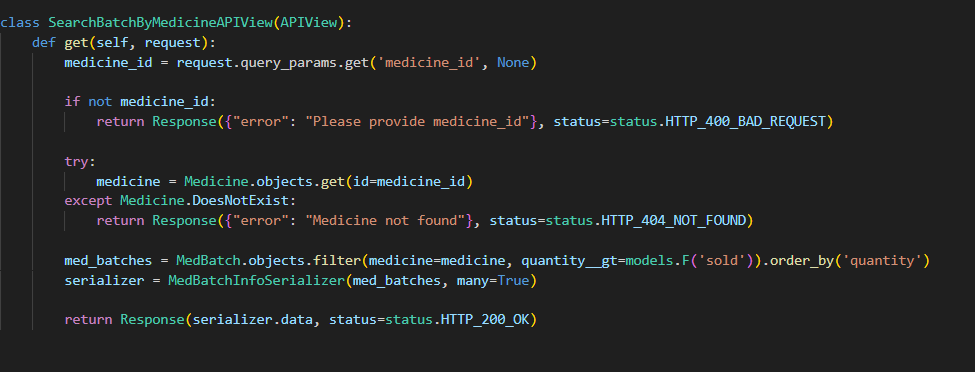


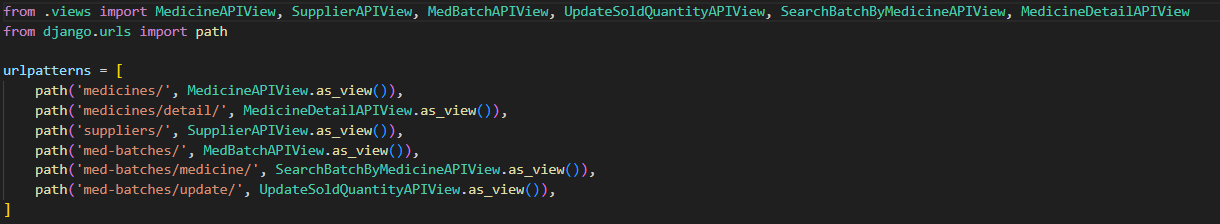




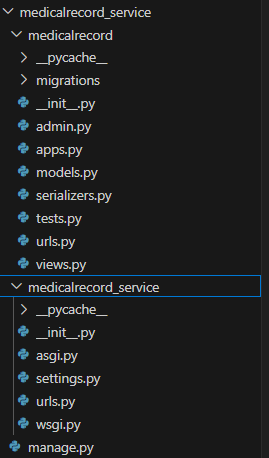


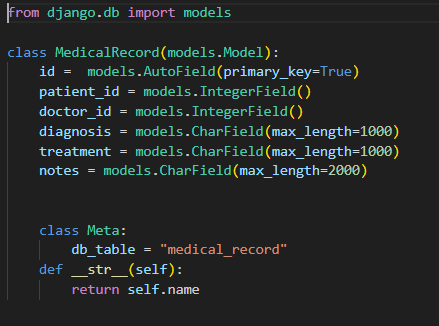




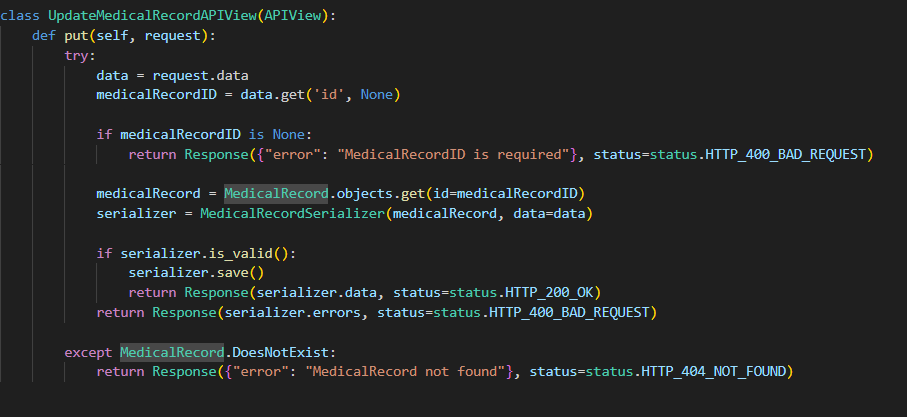


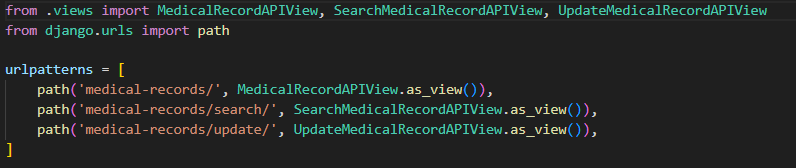
Medicalrecord\_service:



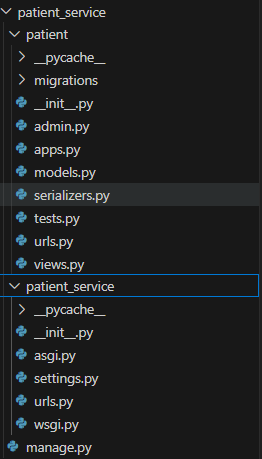


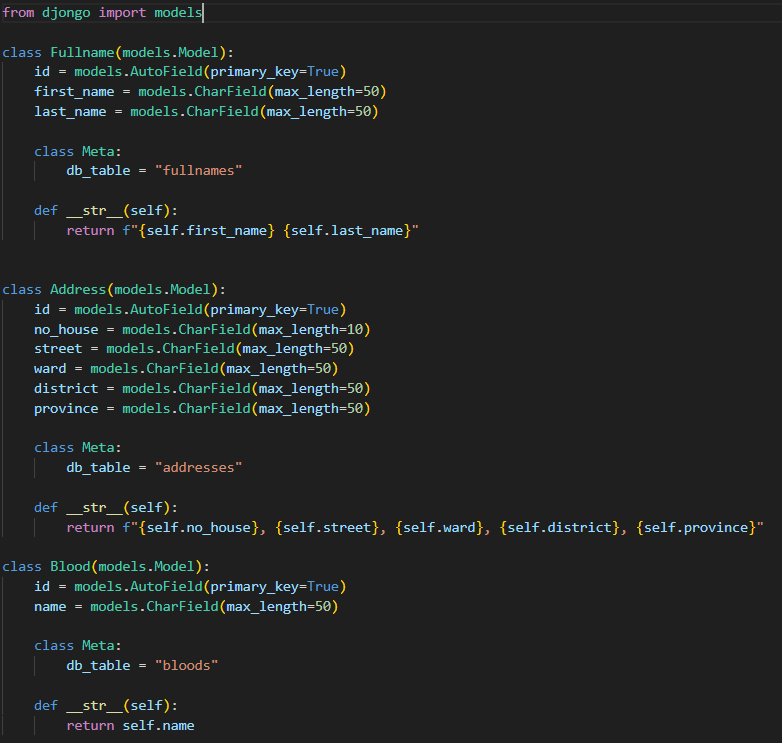


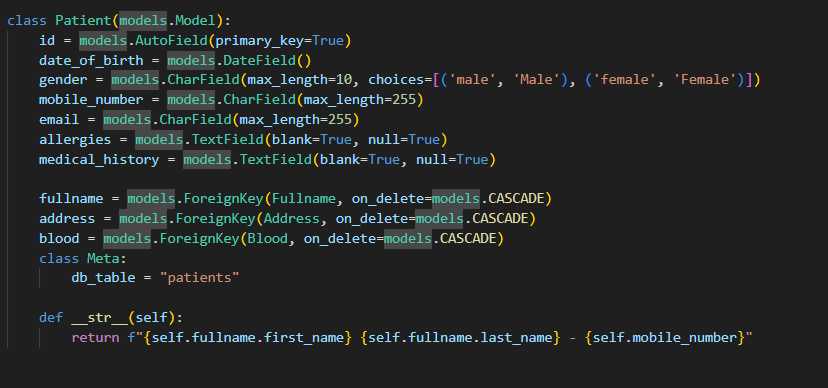


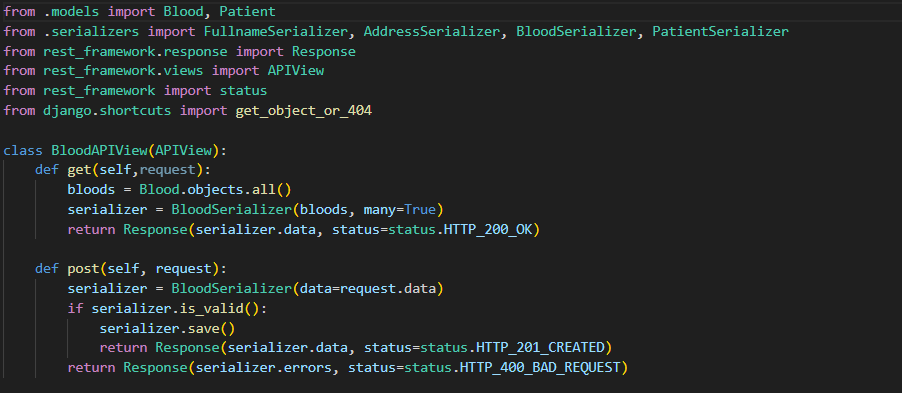


Patient\_service:

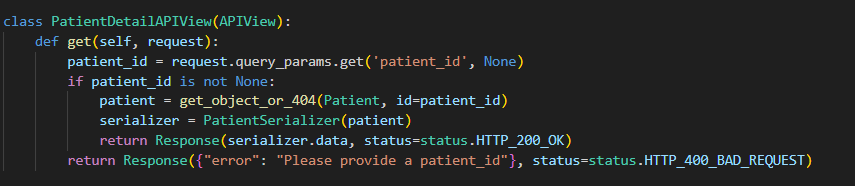




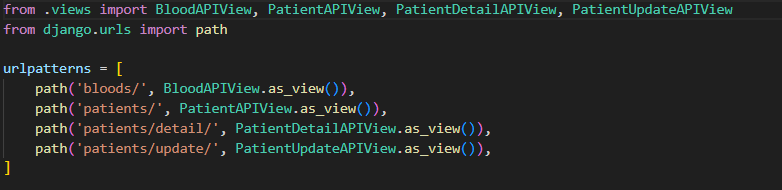












Staff\_service:



